ROBOTIC TANKS • ROBOTS WHO SEE • COMBAT ZONE

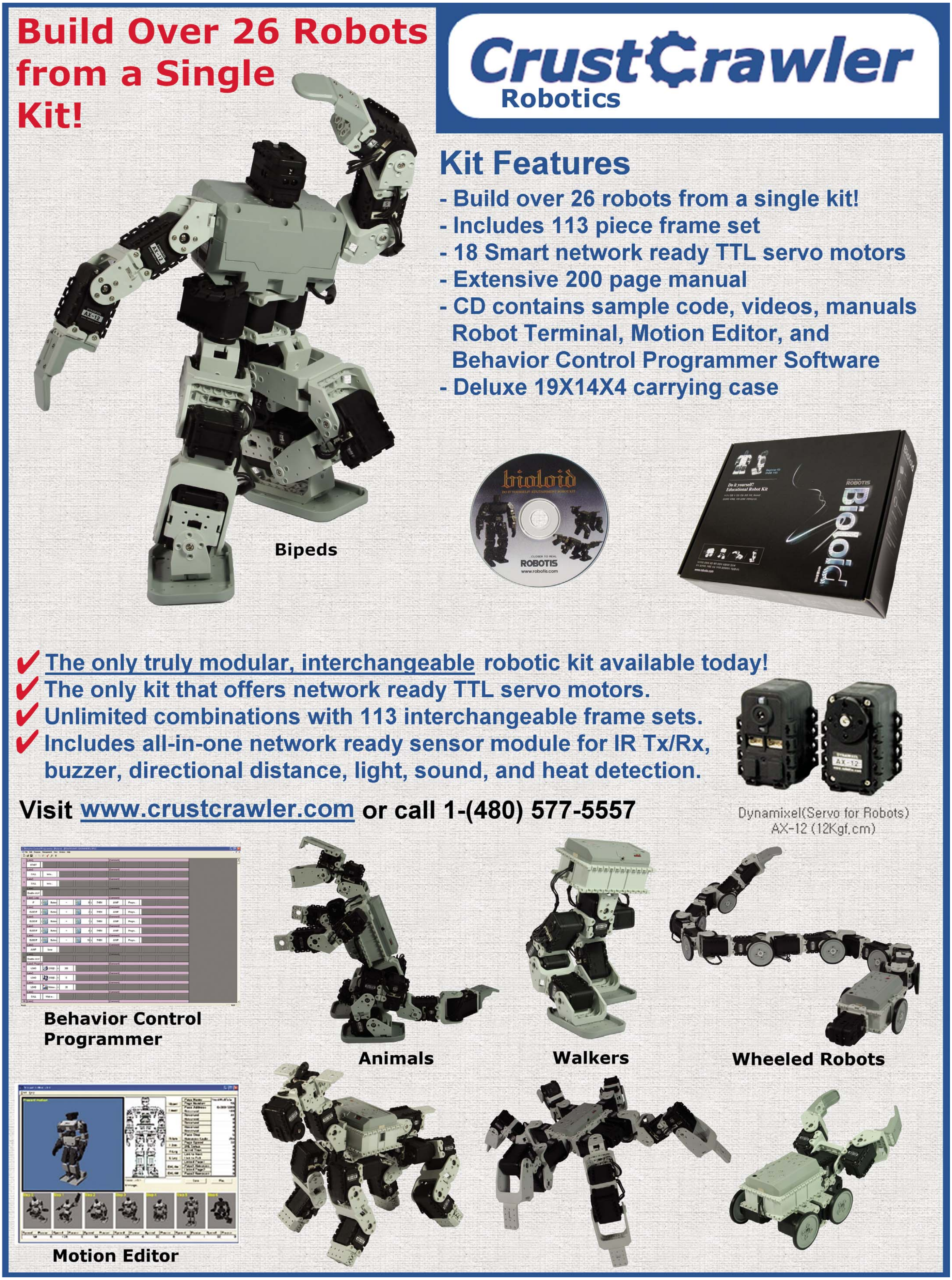
SERVO MAGAZINE

FOR THE ROBOT EXPERIMENTER

www.servomagazine.com
OCTOBER 2006

DIY MARS ROVER

- Energy Management for Autonomous Robots
- Robot's Little Helper
- MIT Microbots
- An Interview With Tandy Trower On Microsoft's New Robotics Studio


Build Over 26 Robots from a Single Kit!
CrustCrawler Robotics
Kit Features
- Build over 26 robots from a single kit!
- Includes 113 piece frame set
- 18 Smart network ready TTL servo motors
- Extensive 200 page manual
- CD contains sample code, videos, manuals Robot Terminal, Motion Editor, and Behavior Control Programmer Software
- Deluxe 19X14X4 carrying case
Bipeds
bioloid
ROBOTIS
Bioloid
✔ The only truly modular, interchangeable robotic kit available today!
✔ The only kit that offers network ready TTL servo motors.
✔ Unlimited combinations with 113 interchangeable frame sets.
✔ Includes all-in-one network ready sensor module for IR Tx/Rx, buzzer, directional distance, light, sound, and heat detection.
Visit www.crustcrawler.com or call 1-(480) 577-5557
Dynamixel(Servo for Robots) AX-12 (12Kgf.cm)
Behavior Control Programmer
Animals
Walkers
Wheeled Robots
Motion Editor

SERVO MAGAZINE

Columns

Departments

ENTER WITH CAUTION!

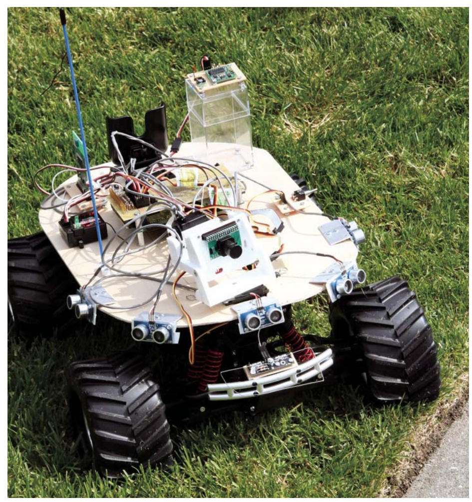
SERVO Magazine **(ISSN 1546-0592/CDN Pub Agree#40702530)** is published monthly for $24.95 per year by T & L Publications, Inc., 430 Princeland Court, Corona, CA 92879. PERIODICALS POSTAGE PAID AT CORONA, CA AND AT ADDITIONAL ENTRY MAILING OFFICES. POSTMASTER: Send address changes to **SERVO Magazine, P.O. Box 15277, North Hollywood, CA 91615** or Station A, P.O. Box 54, Windsor ON N9A 6J5; cpcreturns@servomagazine.com

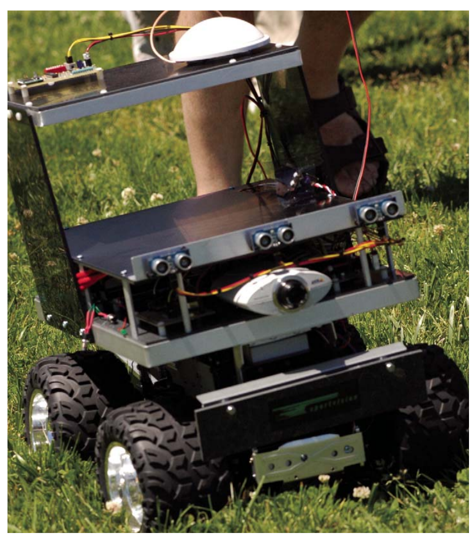
10.2006
VOL. 4 NO. 10

Features & Projects

RoboMagellan Robots come in all shapes and sizes

See Page 62





Mind / Iron

by Dave Calkins

So, you wanna build a robot?

If you've been a reader of *SERVO* for the last year, you've probably followed my series about robot competitions around the globe. From Vienna to Tokyo, and around the US. An incredible array of robot competitions are happening around the world. In my travels, I'm very fortunate to have met literally thousands of robot builders. Men and women, children, adults, and retirees. They come from every walk of life: artists, engineers, doctors, lawyers, cabinetmakers, plumbers, students, programmers, and genuine nut cases. They build all kinds of robots: combat, soccer, sumo, walking, crawling, rolling, autonomous, tele-operated, home-brew, kits, and CAD designed. As a whole, all these varied robot builders have only two things in common:

1) They like building robots.
2) They're shameless procrastinators.

You, dear reader, are in all likelihood one of them. (Now, now. Don't lie about it. I can read your mind through a thin strip of ESP wire in *SERVO*'s cover, which transmits your thoughts to me via a complex RFID & WiFi technology embedded in the staples. Yes, there it is.) You've been saying it for a while now. "I'll finish that robot soon." Tisk, tisk.

So, what do all the robot builders I've met have in common that sets them apart? A deadline! Yes, a deadline!

No, you can't make your own deadlines. (See, I can too read your mind!)

So, here's the *SERVO* challenge: For the next nine issues, we'll be running a series of articles: "RoboGames Prep." *SERVO* is one of RoboGames 2007's sponsors, and we want you to build more robots in addition to reading the magazine.

Yeah, I know — you do build. So, let's *finish* some robots! How will this time be different? Because, you will have a deadline! June 15th, 2007 to be exact. That's when your robots need to be finished so you can compete in the international event at San Francisco, CA, with thousands of other builders from around the world.

Can't make the event? Yes, you can! You've got nine months to plan and save your pennies for a cheap ticket. But that's not the point. Even if you can't make it, if you plan and follow along with our series of articles, you will have finished a robot — or if you're really enthusiastic, you will have built nine robots!

Robots that you can be proud of. Robots that do stuff. Robots that can compete in events around the world — not just at RoboGames in San Francisco. Events can be found from Seattle to Denver to Hartford to London to Tokyo. Or you can start one in your hometown. Or just impress the neighborhood kids with your completed robot(s).

This month kicks off with one of the hardest types of robots to build: RoboMagellan robots. Autonomous, GPS-guided robots that can navigate by themselves. Kind of like the DARPA Grand Challenge, only without needing to use an actual car.

The next seven articles will cover robot builds in order of complexity. As we get closer to our deadline, the

Mind/Iron Continued →

Published Monthly By
T & L Publications, Inc.
430 Princeland Court
Corona, CA 92879-1300
(951) 371-8497
FAX **(951) 371-3052**
Product Order Line **1-800-783-4624**
www.servomagazine.com

Subscriptions
Inside US **1-877-525-2539**
Outside US **1-818-487-4545**
P.O. Box 15277
North Hollywood, CA 91615

PUBLISHER
Larry Lemieux
publisher@servomagazine.com

ASSOCIATE PUBLISHER/ VP OF SALES/MARKETING
Robin Lemieux
display@servomagazine.com

CONTRIBUTING EDITORS
Jeff Eckert, Tom Carroll, Pete Miles, David Geer, Jack Buffington, R. Steven Rainwater, Gordon McComb, Michael Simpson, Ron Hackett, Kevin Berry, Dave Calkins, Phil Davis, Bryan Bergeron, Dan Gravatt, Roger Gilbertson, James Isom, Charles Guan, Pete Smith, Michael Rogers, Wendy Maxham, Russ Barrow, Eric Scott, Michael Mauldin

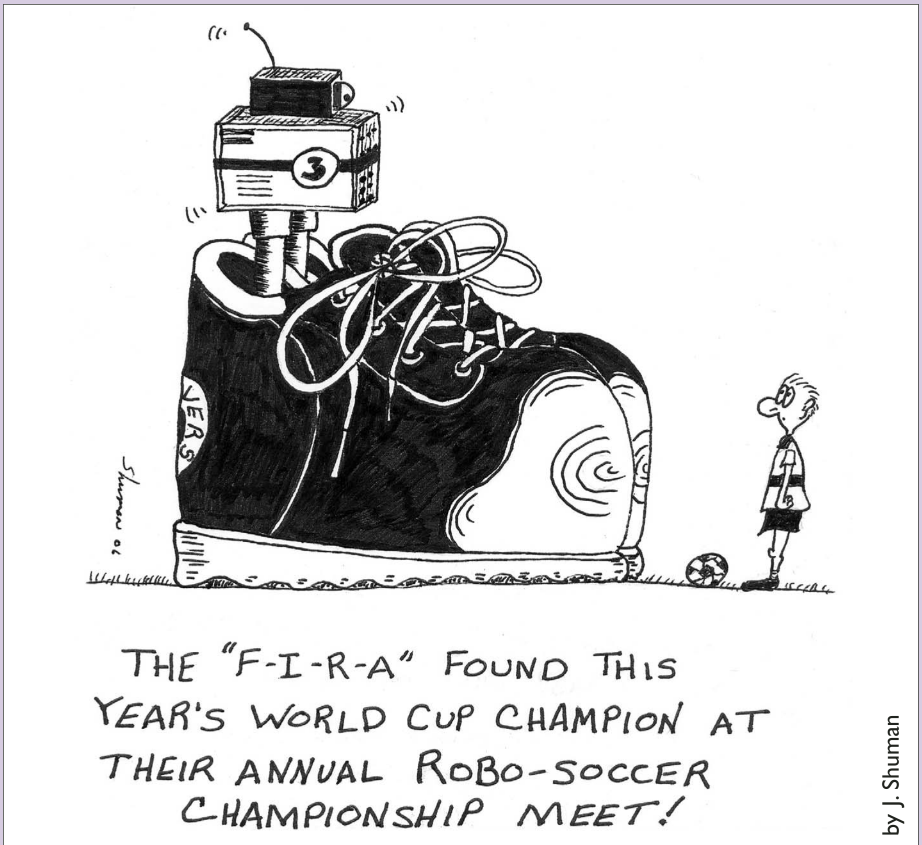
CIRCULATION DIRECTOR
Tracy Kerley
subscribe@servomagazine.com

WEB CONTENT/STORE
Michael Kaudze
sales@servomagazine.com

PRODUCTION/GRAPHICS
Shannon Lemieux

ADMINISTRATIVE ASSISTANT
Debbie Stauffacher

BIO-FEEDBACK



by J. Shuman

Dear SERVO:

I'm a Ph.D. student in computer engineering, and almost every issue of *SERVO* has an article that's relevant to my research. One day I thought, "Wouldn't it be great if I could store these articles on my computer?" That would make it easier to organize and read them. That's when I went to your website and discovered *SERVO* Online. Let me tell you, this thing is fantastic! Not only do you provide a fully searchable database of your archives, but you also have high-resolution PDFs of every issue! I wish all magazines would provide their subscribers a service like yours. Yes, some offer downloadable reprints of articles, but they're usually poor-quality HTML conversions. You provide PDFs of the real thing! I just wanted you to know it's greatly appreciated. Thank you!

Trevor Harmon
University of California, Irvine

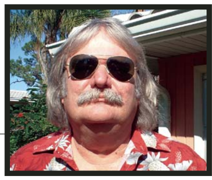
robots will get easier. (Yes, that will help you procrastinate, I know ...) The excellent monthly coverage of Combat Zone will get all you fighters ready, so we'll be covering many other types of competitions individually. Future articles will cover: Androids, which include soccer, Robo-one, and walkers (Nov), Tetsujin (Dec), Fire-Fighting (Jan), Balancer Race (Feb), Art Bots (Mar), Sumo (Apr), and Hockey Bots (May).

You can build any one of these robots and make them competitive. The articles will not give you step-by-steps on making a robot, but they will give you enough pointers for you to be able to make a good start of it and then figure the rest out on your own. No human athlete coasted to a gold medal, and neither will you.

Use your mind. Bend the iron. Make a bot. Show it off.

You can do this. But the clock is ticking. You have nine months left.

I'll see you in San Francisco. **SV**



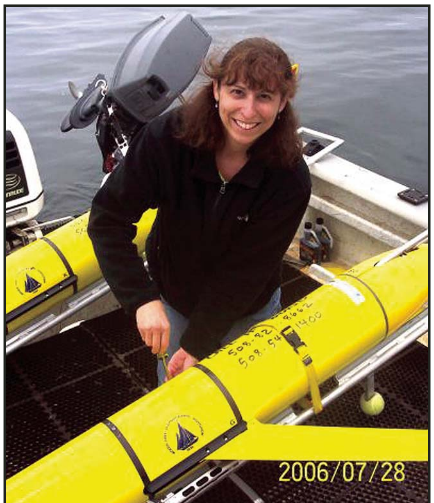
Robytes

by Jeff Eckert

Are you an avid Internet surfer who came across something cool that we all need to see? Are you on an interesting R&D group and want to share what you're developing? Then send me an email! To submit related press releases and news items, please visit www.jkeckert.com

— Jeff Eckert

Automated Gliders Patrol Monterey Bay



Naomi Ehrich Leonard — co-leader of a field experiment of automated undersea gliders — prepares a glider for launch into Monterey Bay, CA. Photo by David Benet.

Aquatic robots are not much of a novelty these days, but in August, some 15 undersea gliders that choreograph their own movements — believed to be the first to do so — were launched into Monterey Bay, CA. The gliders, using mathematical algorithms devised by Princeton's Naomi Ehrich Leonard were programmed to move in a series of rectangular patterns, but the algorithms allowed the gliders to make independent decisions on how to alter their course while moving through a 20 km wide, 40 km long, and 400 m deep area.

The specific purpose of sending the school of fishbots out was to collect information about an upwelling of cold water that occurs every year near Point Año Nuevo, northwest of Monterey Bay. However, the project may lead to the development of robot fleets that forecast ocean conditions and help protect endangered marine animals, track oil spills, and guide military operations at sea.

Two types of gliders — Slocum and spray gliders — were used to take the ocean's temperature, measure its salinity (salt content), estimate the currents, and track the upwelling. The August field experiment is the centerpiece of a three-year program known as Adaptive Sampling and Prediction (ASAP), which is funded by the Office of Naval Research (**www.onr.navy.mil**).

In addition to gliders, the ASAP ocean-observing network includes research ships, surveillance aircraft, propeller-driven vehicles, fixed buoy sensors, and coastal radar mapping. For details, visit **www.princeton.edu/~dcsl/asap/**.

Robot With a Ball

Creator Ralph Hollis (left) and researcher George Kantor are paid a visit by "Ballbot" in the CMU Intelligent Workplace. Photo courtesy of Carnegie Mellon Robotics Institute.

Billed as representing a "new paradigm in mobile robotics" is the "Ballbot," created by Carnegie Mellon's (**www.cmu.edu**) Professor Ralph Hollis. The self-contained, battery-powered omnidirectional unit balances and moves on a single ball rather than legs or wheels, thus allowing it to maneuver in tight places where other bots cannot tread. Although it resembles some sort of strange gyroscope, the machine actually performs its balancing act using an onboard computer that reads information from internal sensors and activates rollers that move the ball, making it essentially an inverse mouse-ball drive.

Ongoing research is aimed at proving that dynamically stable robots (as opposed to traditional statically stable ones) like Ballbot can outperform their static counterparts. Because traditional mobile robots depend on three or more wheels for support, their bases are generally too wide to move easily among people and furniture. They can also tip over if they move too fast or operate on a slope. In theory at least, the concept could lead to robots that more easily move around and interact with people.

Looking for Au in PNG

The Autonomous Benthic Explorer (ABE) — one of two unmanned vehicles used to explore and map hydrothermal vent sites near Papua New Guinea. Photo by Woods Hole Oceanographic Institution.

Meanwhile, in the "nice work if

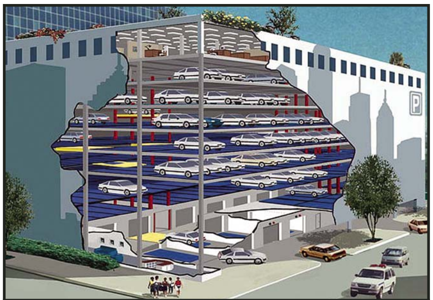
you can get it category," an international team of scientists recently took a cruise to Papaua New Guinea to test out the idea of using unmanned vehicles (both remotely operated and autonomous) to search for copper, gold, and other valuable materials in underwater hydrothermal vents. The cruise is a joint expedition between Woods Hole Oceanographic Institution (WHOI, **www.whoi.edu**) and Canada's Nautilus Minerals, Inc. (**www.nautilusminerals.com**), a mining company that holds exploration leases in the Bismarck Sea within the territorial waters of Papua New Guinea.

Nautilus is the first firm to commercially explore the ocean floor for economically viable massive sulfide deposits and is interested in understanding the size and mineral content of the sea floor's massive sulfide systems. The 42-day trek was headquartered aboard the research vessel Melville, operated by the Scripps Institution of Oceanography (**sio.ucsd.edu**).

Melville is a modest little dinghy 279 ft. in length, with a bit over 4,000 sq. ft. of main deck working area, plus 2,600 sq. ft. of lab space, run by a crew of 23 and able to house up to 38 researchers. She burns 3,600 gallons of fuel a day, so we can hope that the voyage turned up a fair amount of precious metals.

Robo Parking Lots: Boon or Boondoggle?

An interesting concept in automation is offered by Robotic Parking Systems, Inc. (**www.roboticparking.com**), based in Clearwater, FL. The company offers systems for as few as 10 cars on up to more than 5,000, and the installations can be above or below ground, inside or atop a building, or even under a building. On the positive side, the system eliminates parking attendants (and associated tips), saves space, and makes it unnecessary to find your own parking slot; you just drive up into an entrance area, get out of the car, and push a button. The parking system does the rest.



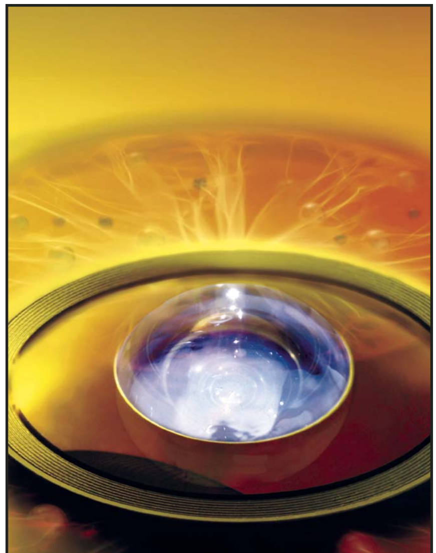
The RPS 1000 robotic parking system can accommodate from 200 to more than 5,000 cars. Photo courtesy of Robotic Parking Systems.

It also largely eliminates the risk of damage or theft, because humans remain outside the garage. On the other hand, because there is no alternative way to retrieve a car, there could be some obvious problems in case of a system breakdown, power outage, or software glitch. In fact, a recent news report revealed that one installation — the Garden Street Garage in Hoboken, NJ — trapped hundreds of cars for several days.

Apparently, the city owns the garage but not the software that runs it, and when the use contract expired, so did the control program. After a short trip into the court system, the city agreed to pay $5,500 per month for a three-year license. But it still might be safer to find a space on the street.

Lenses Feature Autonomous Focus

Inspired by the eye structure of a common fly, a University of Wisconsin-Madison (**www.wisc.edu**) professor has developed a lens that is capable of adapting focusing "from minus infinity to plus infinity" without any external control. Using a hydrogel (a jelly-like polymer) instead of glass, the lens responds to physical, chemical, or biological stimuli to bulge or depress, thus changing its focal length.



Artist's rendering of the smart liquid microlens. Image by Ryan Martinson, Silverline Studio and courtesy of University of Wisconsin-Madison.

The lenses are very small (hundreds of micrometers to about one millimeter), making them potentially useful for lab-on-a-chip technologies, medical diagnostics, detection of hazardous chemical or biological substances, and other functions. For example, when employed with appropriate electronics, one could attach one or a cluster of them to a catheter to provide a peek inside a patient, providing useful diagnostic data or conceivably delivering feedback to a robotic probe.

The technology is being patented through the Wisconsin Alumni Research Foundation, so commercial applications may not be far off. SV

GEERHEAD

by David Geer

Contact the author at geercom@alltel.net

Look Out, Mel Brooks, MIT is Making Space Balls!

Microbots to Explore Mars and Other Space Bodies

MIT has a new idea for robotic, otherworld exploration. By unleashing hundreds or thousands of tiny, redundant, expendable robotic spheres onto and beneath the surface of planets, moons, and stars, MIT hopes to accomplish in-depth analysis of extraterrestrial terrains.

MIT's mobile space balls — dubbed Microbots — will explore crevasses, caves, and perhaps even empty beds where bodies of water may once have flourished. These small, precise, redundant, and cost-effective units — research supported by NASA — would insinuate themselves into every aspect of foreign landscapes.

Size and Motion Help Sensors Get a Notion

Microbots' size and numbers make for efficiency because they will be able to collect data everywhere at the most minuscule levels. They will insure reliability because the destruction of one bot would not affect the performance of hundreds or thousands of others that would easily regroup. They would insure validity because the several bots would be doing many data collections that will have checks and balances against each other.

The approximately centimeter-sized microbots (in one example), could conceivably be launched from an orbiting space vehicle. The balls would initiate typical ball-like movement on their own for mobility, including rolling, hopping, and bouncing around.

The mini-bot's motor skills would be empowered by polymer actuators that would act like little robot muscles.

Rather than using gears, gearboxes, and grease, the microbots use elastic materials that flex in an orthogonal manner to move the bots around. This elastic method of movement uses many times fewer parts that are much lighter and don't rub together to create wear and tear.

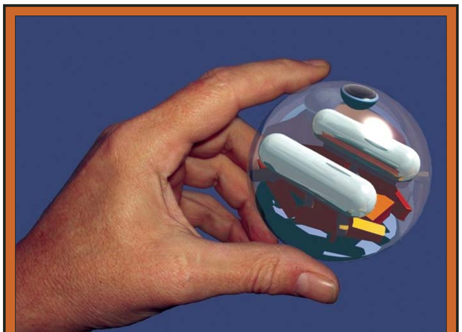
However, this elastic "motorvation" is slower than gears and motors. To resolve this, the elastic actuators store energy over time and release it quickly to create their quick jumping motion.

Robot sensors will include imagers, spectrometers, sampling devices for soil, and other materials samples and chemical detection sensors.

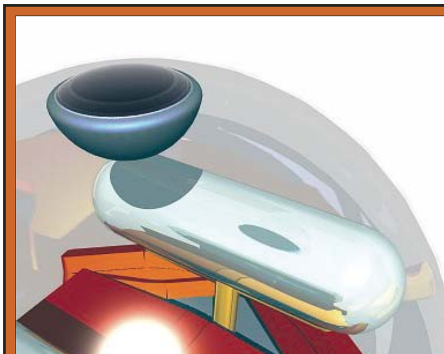
These sensors are used to assess soil, topography, and the constitution and anatomy of rocks. Microbots will work in tandem as a network, distributing information among themselves to analyze the larger picture of what they each see individually.

Illustrations are by Gus Frederick.

Drawing of a conceivable, baseball-sized microbot. Actual microbots may be much smaller.

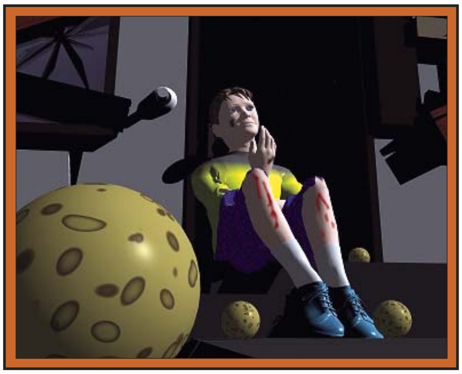


Closeup of ballbot — baseball-sized probe — drawing.

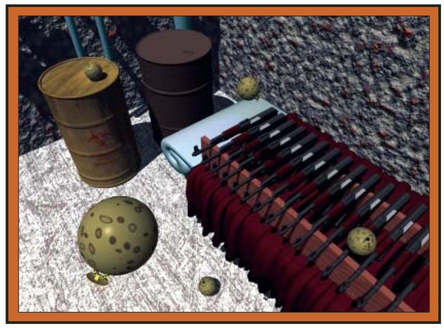


Artist's rendering of a much smaller probe.

Artist's rendering of a camouflaged baseball-sized probe.



Drawing of a room full of camouflaged probes with a sitting child.



Camouflaged probes in an armory. These drawings are quite interesting considering that I found no mention of military uses for these probes.

Probe Communications and Construction

The probes will navigate both icy and hot surfaces, sending sampled data to a lander or spacecraft via low power radio waves. The robots would also communicate with each other over makeshift wireless LANs. This will enable them to share information despite their being spread out in caves and other areas. Their sheer numbers would make for valid and reliable data collection.

Microbots will be made of transparent polycarbonate balls. The balls will be equipped with actuators, fuel tanks, cameras, sensors, and sniffers.

Microbots will hop, bounce, and roll into position, in that order. Because of its precise weight, the ball will roll into a standing position on its single foot after one roll.

Communications will be transmitted between and through the bots back to a lander, which will transfer the data to the orbiter or back to Earth. Microbots will need to use communications for navigation, to determine their location relative to each other and to communicate information about their surrounding environment to get a bigger picture of the landscape. Each bot comes equipped with a transceiver to accomplish this.

Surface missions can be accomplished over an area of about 135 square kilometers. Such missions require no more than 1,000 bots. For these missions, the bots would communicate over higher frequency radio. In research, 31 GHz has been optimal. The bots would also use miniature phase array antennas. The bots will also be equipped with miniature data processors, 4 GB of disk space.

Communications for the movement of the microbots may be accomplished by decentralized systems or virtual pheromones (discussed in another GeerHead). Instead of having a central unit of control, the robots would share control over the team as if moving as a herd.

Scientists believe that the bots could collect and transmit several MBs of data daily. This requires onboard data processing. Future computing capabilities should arrive in time to more than meet these needs.

They Need Fuel

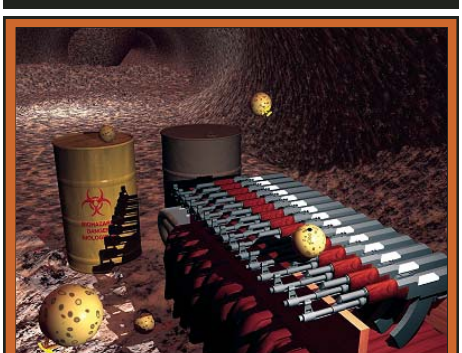
By using a special hydrogen/oxygen micro fuel cell, the bots will be able to hop and take in data for about a month on an average mission. The fuel cell concept comes from Stanford U. These fuel cells can generate a lot of energy for their size at lower power rates. The energy is stored in the plastic foot mechanism of the bot.

The bots need to make about one hop per minute to accomplish their missions. So, the fuel cells can produce enough energy for the hopping mechanism to store data just in time to hop.

The fuel cells are much smaller and more efficient than conventional batteries. Using these cells, the bots can complete the 5,000 jumps necessary for their missions, after which the bots are obsolete and simply stop moving.

The fuel cells don't power the jump-

Probes in a cave-based armory.

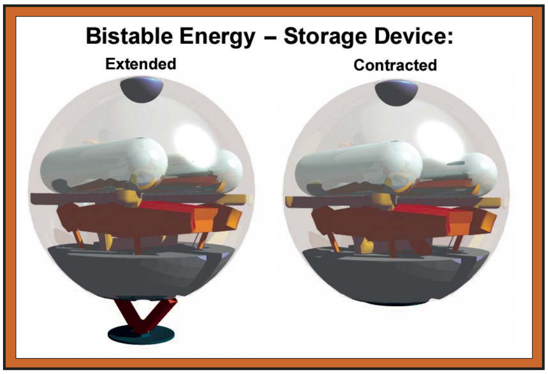


SENSORS

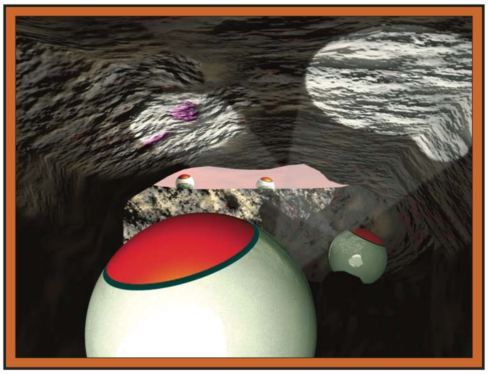
The microbots will use sensors to collect geochemical data for analysis. This includes basic chemical data, geophysical data, geothermal data, climate data, rock and mineral data, and organic data. Sensors will also detect methane, microbes, organic molecules, sulfur compounds, and water. Sensors will also measure temperature and pressure.

Sensors will have uses besides data collection. They will need to use them to navigate; to determine their locations and movements. They will also use accelerometers and gyroscopes.

The bots will also use panoramic imagers — cameras that take a panoramic view — to identify sites of interest for further investigation. A microbot may be capable of carrying two imagers so as to provide stereo images.



Side-by-side images of the probe with its foot extended and then retracted.



Probes with "headlights" make their way through a cave.

ing mechanism alone. They also power sensors, communications, and microcomputers. But, all systems require the same or less wattage and none would be running at the same time.

It All Adds Up

By combining these features, researchers can drive the bots into very difficult terrain over long distances. Through studies, researchers have shown that the bots can jump 1.5 meters high and move one meter horizontally, even under Martian gravity (Mars is one of their target explorations for these bots).

Microbots can dig into uphill and downhill loose dirt so as to maintain their position. The bots low gravity center helps it maintain its standing position against most odds. Even if it rolls over it will end up on its foot — more talented than a cat, don't you think?

The microbots are expected to spread out to investigate up to 50 square miles as a team, all within 5,000 (number not distance) expended hops.

Challenges

Microbots may get trapped where lava has broken down, trapped between pieces of lava if they hop into them. At the same time, researchers are not certain that the breakdown piles will still exist to such a degree that they will cause such problems.

The microbots may also face communications challenges in caves. Considering factors such as the use of short-range radio and the fact that radio waves will be absorbed into the rock of the caves, communications can be muffled or hampered. So, the bots will have to communicate with each other along a LAN made up of Microbots, leading out of the cave to get clear communications from the bots outside the cave to the land vehicle or the orbiter.

Probes on an icy landscape.

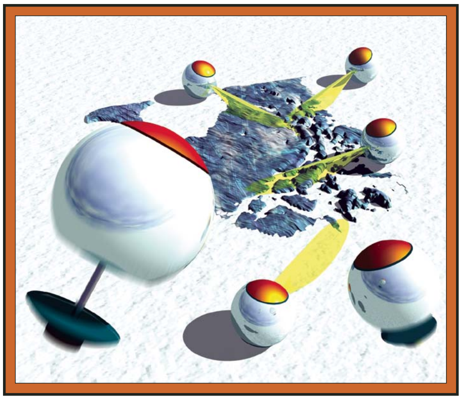
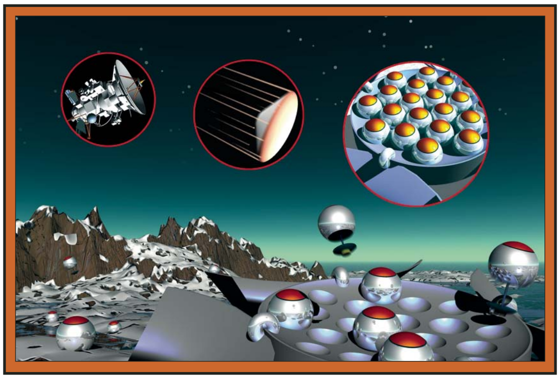
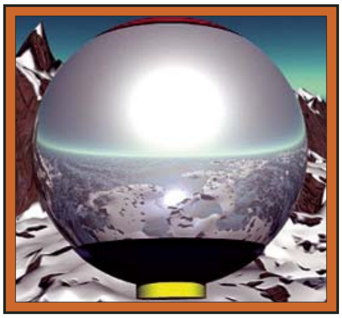


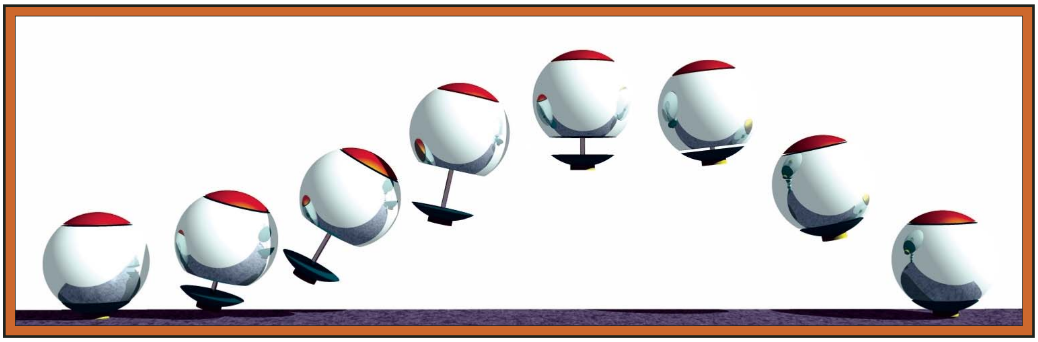
Illustration of potential probe delivery methods including a probe lander.



Hopping probe in icy environment.



Probe sequencing through a hopping event.



AND MORE SENSORS

Mass spectrometers would be used for chemical sensing; these sensors use magnetic or electric fields to do their sensing. Some may even use radiation to create an electric field. These spectrometers will need to disintegrate the sample using a laser or similar tool and then absorb the sample for study. Research is underway to develop advanced mass spectrometers for this purpose.

To accomplish this, the microbots can be programmed each to stop at various degrees of entry into the cave. In this way, the microbots can relay data out of the cave using 2.4 GHz radio waves.

Conclusion

The microbots are not only promising, but likely the cheapest, most efficient, and accurate means of extraterrestrial data collection for the future. **SV**

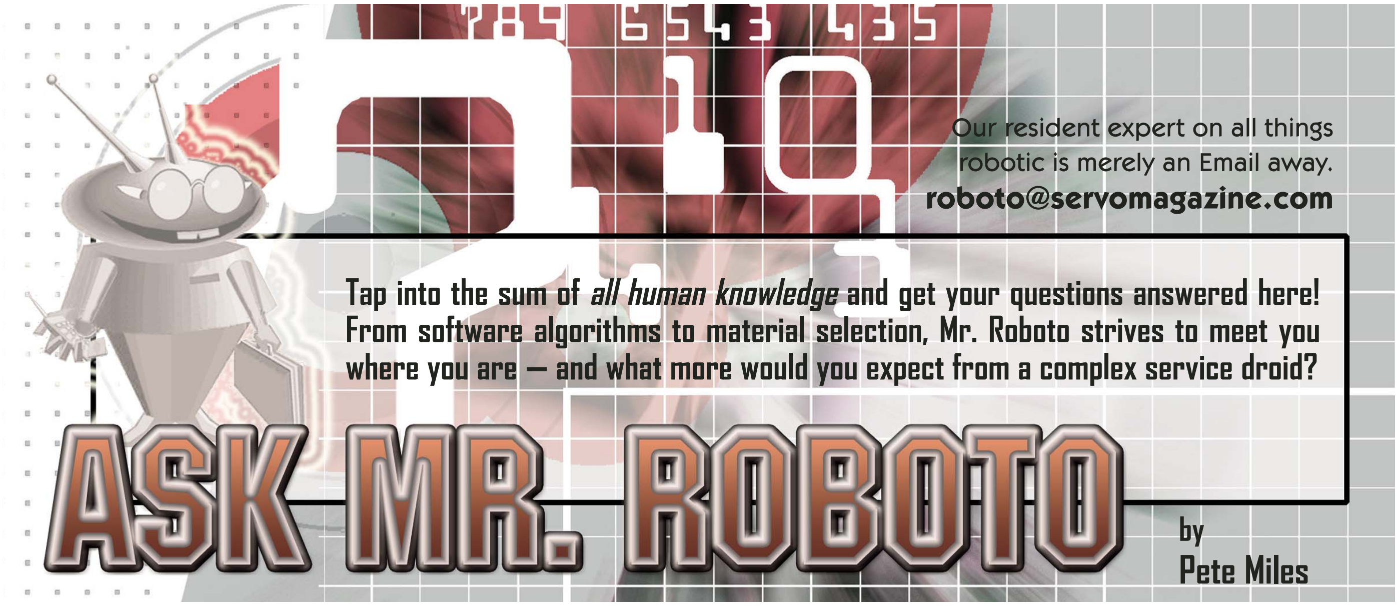
RESOURCES

Microbots page at MIT
http://robots.mit.edu/projects/microbots/index.html

Video of probes
http://robots.mit.edu/projects/microbots/FSRL_NIAC_2004_2.avi

Work behind Microbot probe's foot
http://robots.mit.edu/projects/mechatronics/index.html

NASA division sponsoring Microbots
www.niac.usra.edu



Our resident expert on all things robotic is merely an Email away.

roboto@servomagazine.com

Tap into the sum of *all human knowledge* and get your questions answered here! From software algorithms to material selection, Mr. Roboto strives to meet you where you are — and what more would you expect from a complex service droid?

ASK MR. ROBOTO

by **Pete Miles**

Q. Is there an easy way to change the output from a sensor with electronics instead of using a program? I have some of those Sharp range sensors hooked up to a BASIC Stamp. When the sensor detects an object, my program will turn a red LED on. If I hook up an LED straight to the sensor, it will be on when there is no object in front of it, and will turn off when it sees an object. I would like to have the LED turn on when it sees an object without having to use the BASIC Stamp.

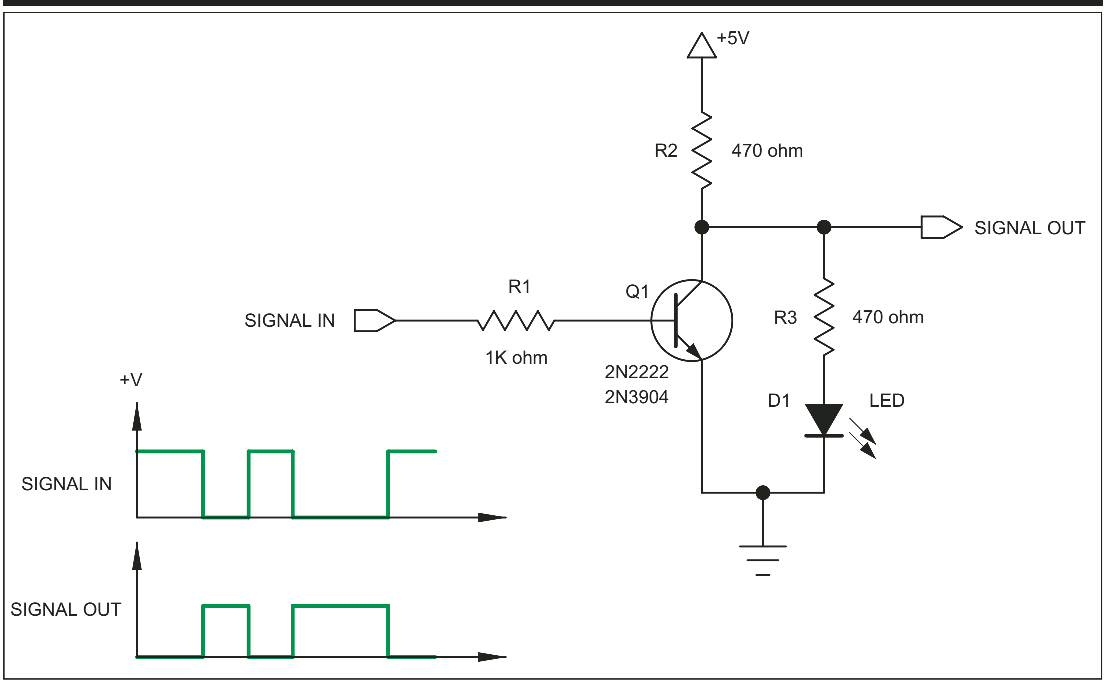
— Jill Verge

A. For many years, I have wondered why many sensors output a high signal when they don't detect anything, and output a low signal when they do. From a failure analysis and safety point-of-view, this is backwards. You would want the sensor to output a high signal if it detects something, that way you will know for sure that it is has. If the output is low when it detects something, you don't know if the sensor is actually detecting something, or if it is broken or defective.

I also have used software to light an LED to provide a visual indication of whether a sensor is detecting an object or not. If you have the available I/O pins on your microcontroller project, this is usually an easy thing to do. However, there are times when this has to be done with hardware (electronics). What you are looking for is called a signal inverter. There are many different approaches you can use to do this, but I'll describe the two I use most in my projects. Keep in mind that this is for digital signals, not analog signals.

The first approach is to use a basic, general-purpose NPN transistor and a couple of resistors. Figure 1 shows a simple schematic using a transistor to invert an input voltage signal, along with a couple of sketches showing how the output signal is inverted from the input signal. For most applications, this circuit will work fine for inverting a digital signal from a sensor. But the output voltage will always be less than five volts (it will be approximately (Vcc + V_F)/2 where Vcc is the five-volt supply voltage and V_F is the LED's forward voltage). If the downstream circuits (i.e., microcontroller) will interpret the lower voltage as a logic 1, then you will be fine. Most

Figure 1. Simple transistor-based signal inverter.



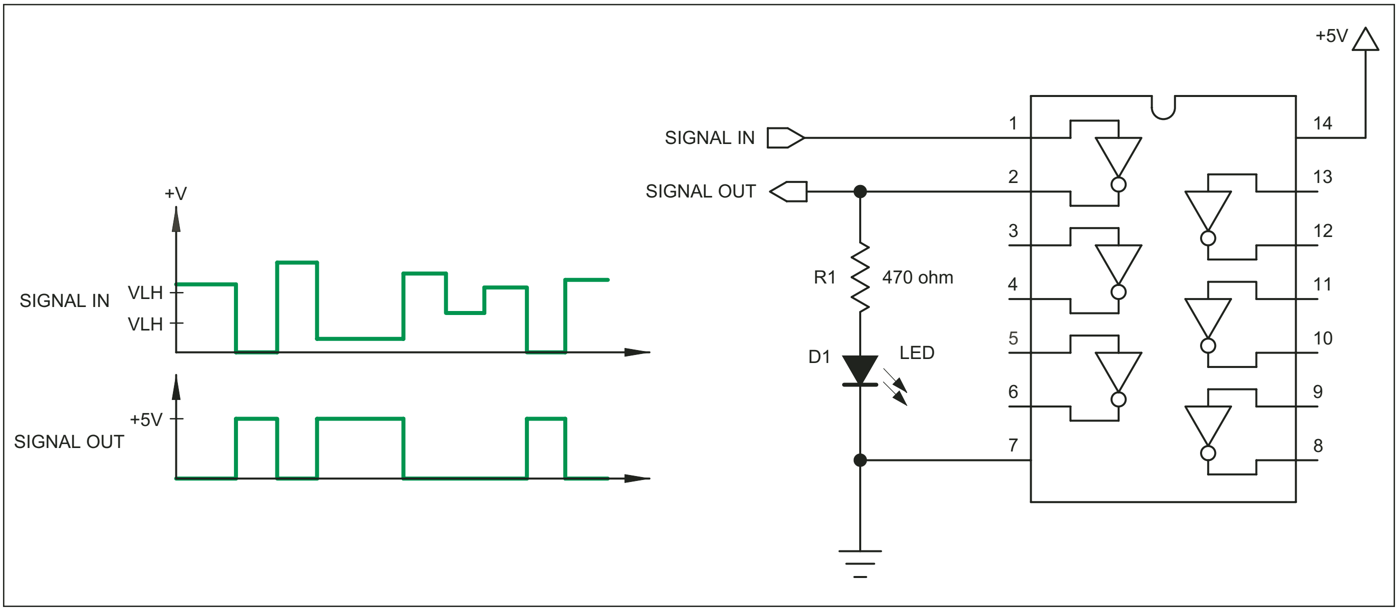


Figure 2. Simple hex inverter-based signal inverter.

LEDs have a forward voltage around two volts, so the output signal will be around 3.5 volts, and most digital circuits will interpret this voltage level as a logic 1.

The second approach that works quite well is to use hex inverters. Since the primary purpose of hex inverters is to invert digital signals, they are ideal for your application. There are many different versions of hex inverters to choose from, such as the 7404, 74C04, and 4069, or the Schmitt version of the hex inverters such as the 74C14 or the 4584, just to name a few. In most cases, it really doesn't matter which one of these you choose for this application; they will all work fine.

Figure 2 shows a simple schematic using a hex inverter to invert a sensor signal and displaying the result with an LED. As you can see, there are fewer components needed to use a hex inverter than using the transistor approach described previously. With a single hex inverter, you can invert six different sensor signals, which will take up less space than using six transistors and 12 resistors (not counting the current limiting resistor and LED pair).

In Figure 1, you should notice that the output signal voltage is constant, but always less than the five-volt supply, but when you use the hex inverter, the output voltage is either five volts or zero volts (see Figure 2). Also, there are certain input voltages that the hex inverter will interpret as a logic 0 or a logic 1. The voltage threshold is different when the voltage is transitioning from a low to a high state (V_{LH}), or from a high to a low state (V_{HL}). This has the advantage that an analog signal could be conditioned into looking like a digital signal. A similar effect can be seen with the transistor approach, but if the voltage gets too low, the output voltage will drop even further.

Both of these approaches will work well at inverting the signals from your sensors, so when they detect an object, they will output a high signal and light an LED, then turn off when no object is present.

Q. Where is a good place to get raw aluminum materials at?

— Mel Forenster

A. Just about any industrial metals supplier will have all the aluminum you would need. They are not your local hardware or home improvement store, though some of them may carry aluminum that would meet your needs. If you are looking for a local source, the Internet is usually not the best place to find it. Instead, the Internet will tell you about metal suppliers from around the world. It is difficult to filter the search down to businesses that are within driving distance.

The best place to look to find your local metals supplier is your phone book. I would first look under Aluminum and see if there are any companies that are listed that way. Sometimes you will see subcategories called Distributors or Wholesalers. These are the places to contact. If there are no businesses listed under Aluminum, then look under Steel. Most companies that sell steel will sell aluminum or, at the very least, tell you where you can get it.

Here is a hint that will save you money when working with your local industrial metal supplier. When you ask them if they have what you are looking for, ask them if they have any remnants that are large enough to fit your material requirements. Remnants are leftover pieces of material from a previous job. They are usually odd sized. Though most companies will sell you the material by weight, even if the remnants are larger in size than you need, the money savings will come in what is called a cut charge. Cut charges can be very expensive — sometimes hundreds of dollars — depending on the tools needed to cut the raw material you need from a larger sheet. Buying the remnants saves you this charge, and in many cases, it will save you money when getting the material.

Many times when you are buying a small piece of material, the cut charge is greater than the raw material costs themselves. Keep in mind that

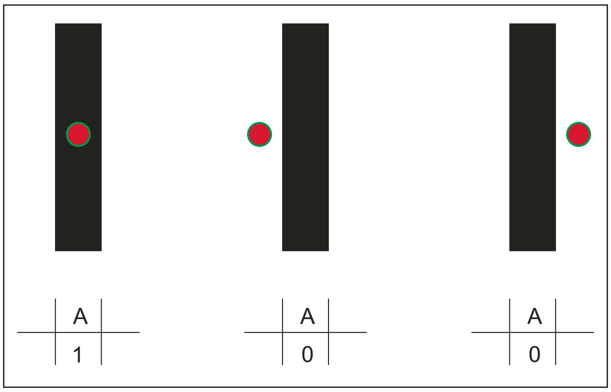


Figure 3. A one sensor, line-following robot, showing how sensor output changes while finding a line.

remnants are leftover material, so they may not be available and you will have to pay for the full cut charge.

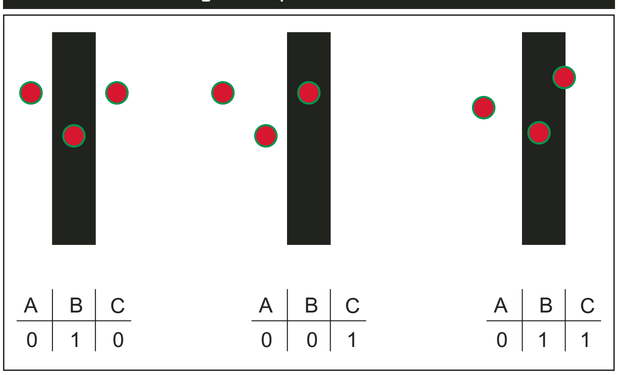
If you are willing to mail-order the material, then there are lots of places on the Internet that will sell the material to you. Two places I like to work with are McMaster Carr (**www.mcmaster.com**) and Metal Supermarkets (**www.metalsupermarkets.com**). Though the costs of the raw materials at McMaster Carr are usually higher than local suppliers, they are convenient and you can get all of the other mechanical hardware you need for your project. McMaster Carr is the engineer's one-stop shopping store. Metal Supermarkets always seem to have what I need, when I need it. With over 80 stores nationwide, it is pretty easy to go and pick up the material yourself to save on shipping charges.

Q. I have a line-following robot that does a really good job at following a line, but I would like to make it go faster. Right now, the robot has a tendency to spin around to find the line when it drifts off of it, or occasionally does a complete 360 when it comes to a right angle corner. My robot uses an IR LED and a phototransistor for the line sensor, and I have it placed between the two wheels of my two-wheeled robot. Is there a better place to put the sensor, or is there a good strategy for figuring out which way to turn? Any help would be appreciated.

— Rob Holder
New York

A. Robots with a single line sensor can be made to work quite well, as you have observed, but there are some uncertainties that occur when the sensor loses track of the line. Figure 3 is a simple illustration of a single sensor being used to tack a line. The red circle is the sensor, and the black line is the line the sensor is trying to track. This illustration is more for people just getting started with line-following robots. The left side of the figure shows the sensor centered over the black line. The output of the sensor is assumed to be a logic 1 (the actual output depends on the type of sensor you are using). The center image and the right image show what happens when the robot drifts off to the left or right side of the line. When the sensor moves off the line, the output changes from a logic 1 state to a logic 0 state (i.e., On and Off the line).

Figure 4. A three sensor, line-following robot showing more position information.



When the robot detects the logic 0 state, it knows it has veered off the line. But as you can see, the robot doesn't know if it is on the left or right hand side of the line, because all it knows is that it is getting a logic 0 output from the sensor. This isn't necessarily a bad thing. If the logic in the robot says to rotate clockwise if it loses track of the line, it will find the line quickly if it veered off to the left, or will rotate 180 degrees if it veered off to the right. I suspect that your one-sensor robot has the tendency to also go the opposite direction from time to time when it loses track of the line.

As with any robot, the more sensor information you can get, the better it can respond to its environment. There is a lot of debate in the line-following robot community as to the best number of sensors to use. Some say three, others say four, five, or seven individual sensors. And then there is even a bigger debate on what is the best placement for the sensors. The more sensors you have, the less critical the exact location of the sensors, but the best placement really depends on the type of lines your robot is expected to follow. Do the lines have gentle curves, right angles, change width or colors, change from solid to dashed lines, etc.?

The other question depends on putting them at the center of the robot or in front of the robot. That really depends on how fast your robot can read the sensors, process the information, and how fast the robot can react. Center of the robot body is fine, so is the front of the robot. I have seen some photos of some Japanese line-following robots, and they have their sensor arrays several inches in front of the front wheels.

To give you an idea of how well multi sensors work, let's take a look at a simple three sensor approach, shown in Figure 4. The left side of the figure shows the sensors centered on the line. The outputs of the three sensors are shown in the truth table below the image. You will notice that the two side sensors are placed slightly forward of the center sensor. (You'll see the advantage of this placement later.) With this configuration, if the robot slowly drifts off to the left or starts turning to the left, the output from sensor C is triggered, which tells the robot that it needs to move/turn back to the right. Sensor A would be triggered if the robot moved/turned/drifted off to the right. With this configuration, your robot will know exactly which side of the line it is on if it drifted off-course.

Figure 5 shows some different line-following situations that three sensors can uniquely identify. The left and center images are right angle line cases. The

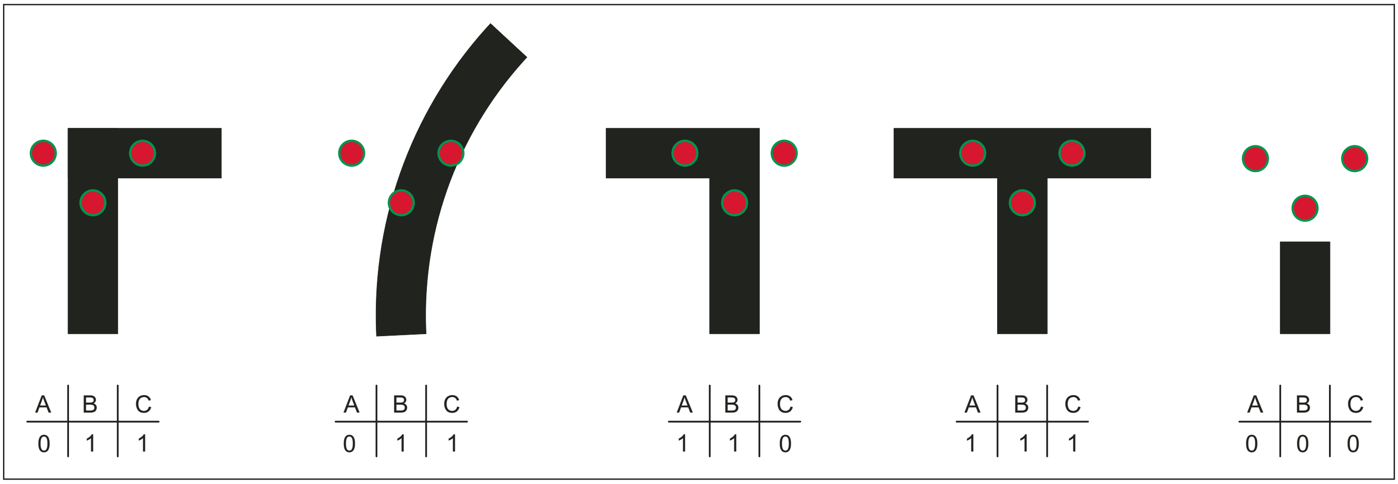
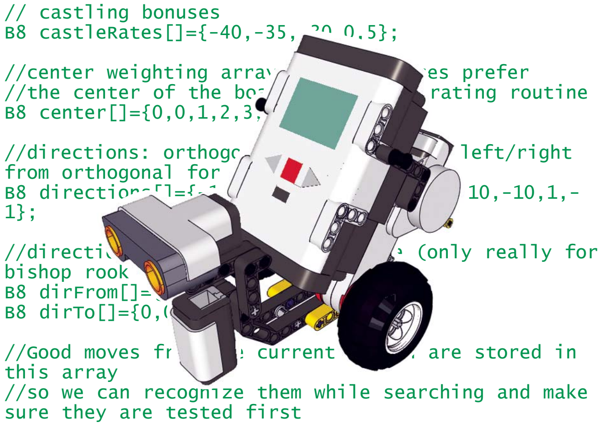


Figure 5. A three sensor, line-following robot coming across multiple line configurations.

side sensors will detect the direction of the right angle. The advantage of having the side sensors forward of the center sensor is so that the robot can start the turning algorithm earlier. The "T" intersection and the "End of the Line" configurations shown in the right two images do become a bit confusing. Do you turn left or turn right in these cases? It is up to you how you want to handle this, but your robot will know that it is at a T intersection or has come to an end of the line. I personally like to use the Right Hand Rule: When in doubt, turn right.

The sketches you see here in Figures 4 and 5 should give you an idea how to figure out how many and how to orient them. Drawing up the various types of line situations and placing your sensor array over them will help you decide what works well for you. It is best if you draw them to scale because the relative distance between sensors may become limiting factors. If the sensors are too far apart, the line could fall between sensors, and the robot could get confused again. SV

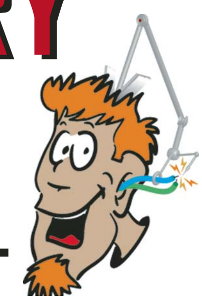


LESSONS FROM THE LABORATORY

A bi-monthly column for kids!

NXT Robotics: First Build

with Brian Davis
by James Isom

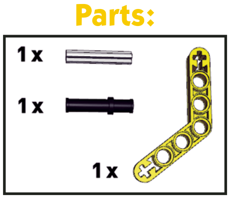


This month, we're going to introduce our first NXT based robot from LEGO robotics guru Brian Davis. I first saw this chassis last August at NIWeek in Austin, TX where we were officially announcing the NXT robotics product line. There were robots from various designers, but I kept finding myself going back to this one for its simple, yet functional design. Brian was gracious enough to agree to let me make building instructions for it and share it with the readers of *SERVO*. So, I bring you ... Jenn Too.

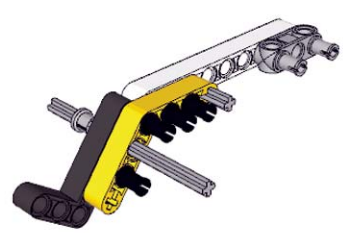
JENN TOO — CHASSIS INSTRUCTIONS

STEP 1:

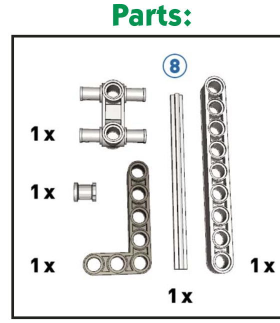
Parts:



STEP 2:

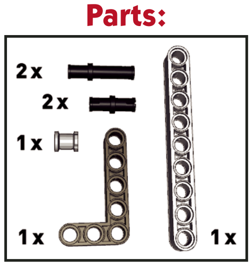


Parts:

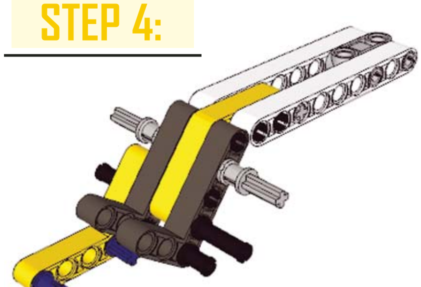


STEP 3:

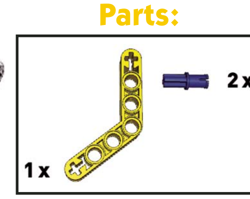
Parts:



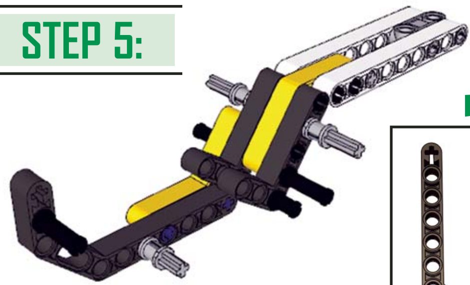
STEP 4:



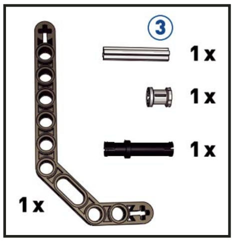
Parts:



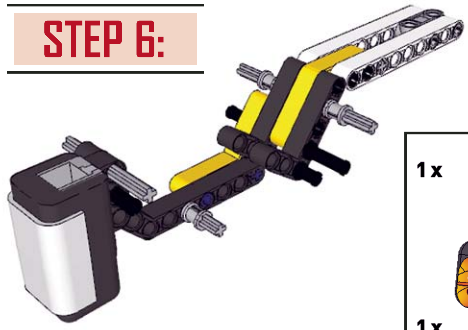
STEP 5:



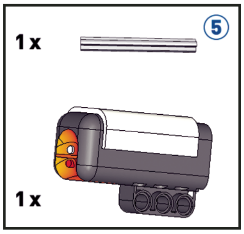
Parts:



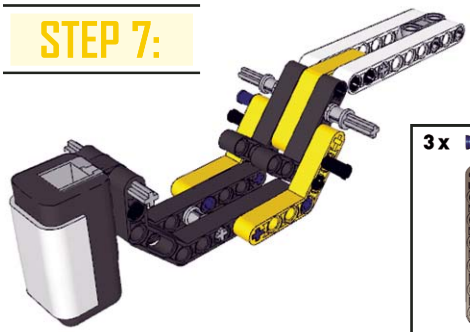
STEP 6:



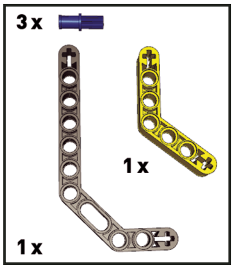
Parts:



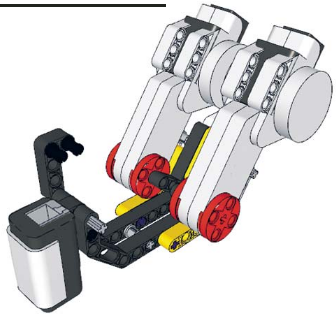
STEP 7:



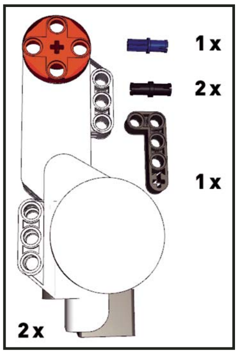
Parts:



STEP 8:

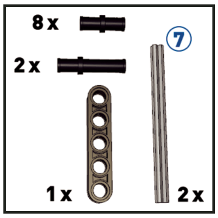


Parts:

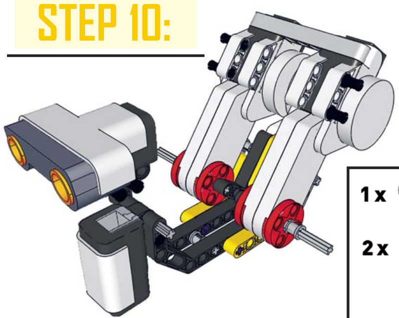


STEP 9:

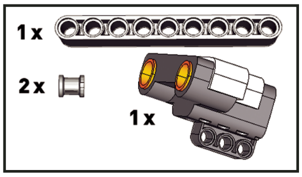
Parts:



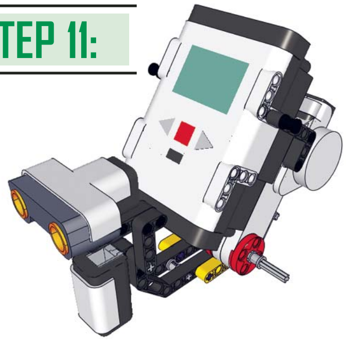
STEP 10:



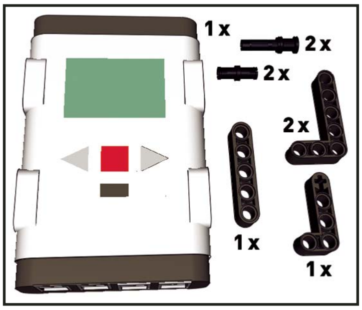
Parts:



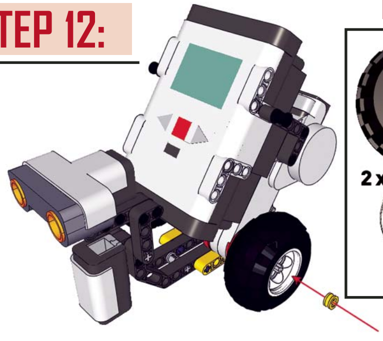
STEP 11:



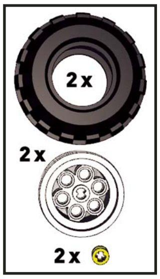
Parts:



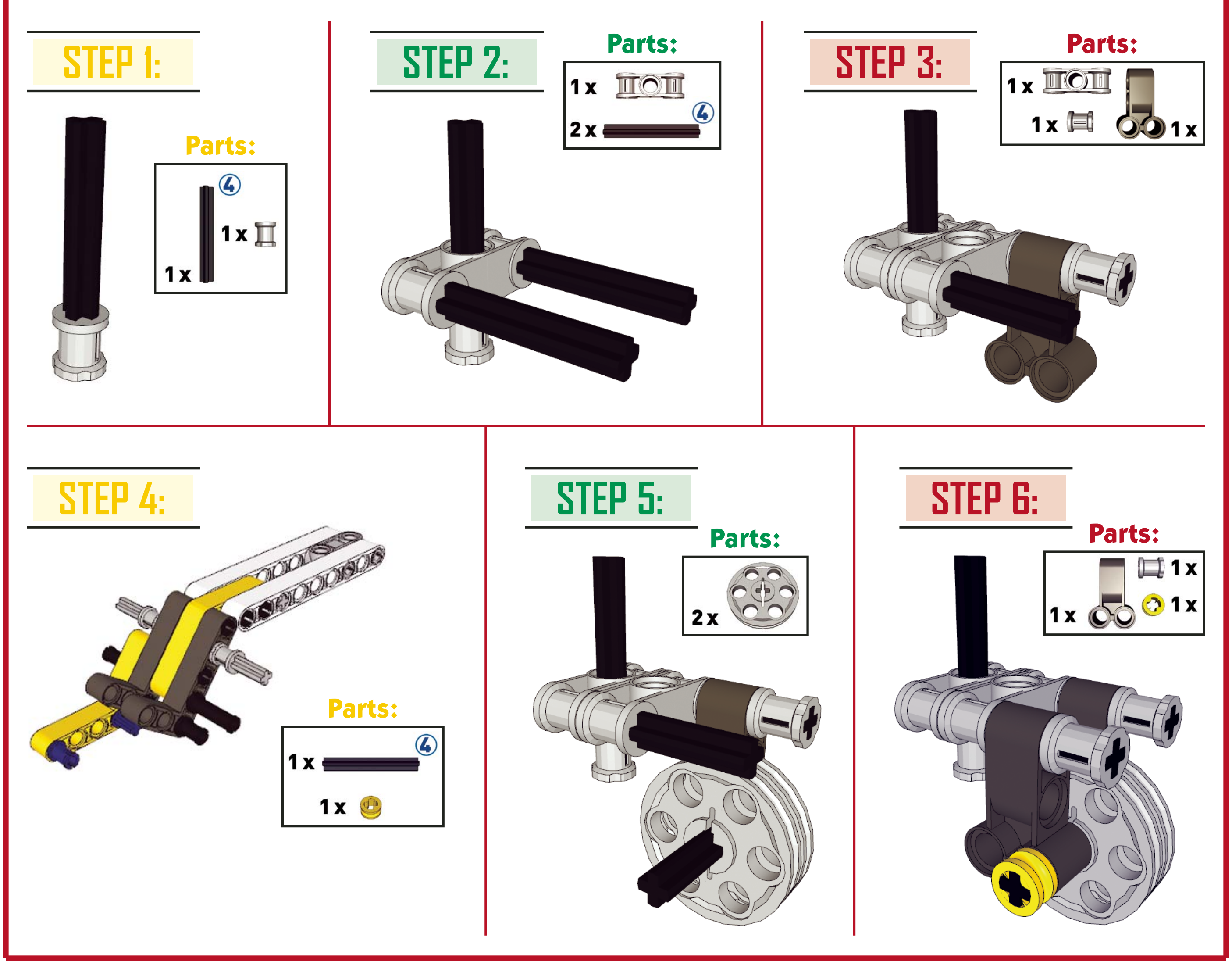
STEP 12:



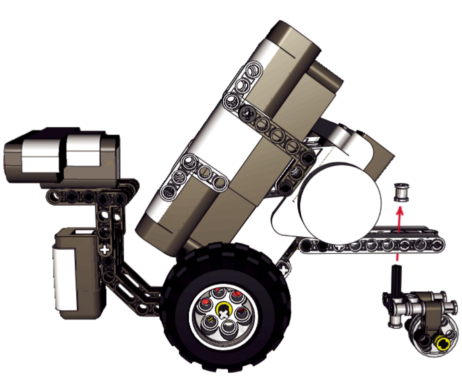
Parts:



JENN TOO — CASTER WHEEL INSTRUCTIONS



JENN TOO — GRAND FINALE

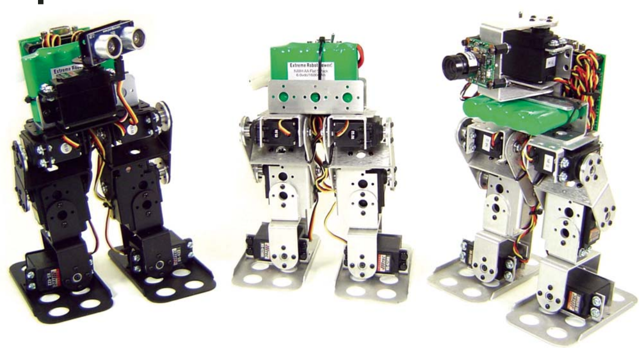


That's it! You're all finished. In the next issue, we will build and program Brian Davis' remote control for Jenn Too. Until then, happy building. **SV**

NEW PRODUCTS

CONSUMER ROBOTS

Biped BRAT



Lynxmotion introduces the all new Biped BRAT — a Bipedal Robotic Articulating Transport that costs less than $200. A full kit including SSC-32 servo controller and Visual Sequencer software is available for less than $300.

The BRAT is a simple six-servo biped walker featuring three degrees of freedom (DOF) per leg. Even though it only has six servos, it can walk forward, backward, and turn in place with variable speed. It can even get up from lying on its front or back. The BRAT can also do acrobatic-style moves. See the Lynxmotion website for a video gallery.

The robot is available with brushed or black anodized aluminum servo brackets from Lynxmotion's Servo Erector Set. It is fully compatible with the SES so you can expand the robot as your skill level and/or budget allows. Getting the robot moving with the Visual Sequencer is easy because there are 10 sample routines included. The powerful database-driven program supports importing and exporting projects, so you can share your cool moves with other users. The program exports Basic Atom and BS2 code for autonomous operation.

For further information, please contact:

Lynxmotion	Website: **www.lynxmotion.com**

CONTROLLERS & PROCESSORS

CUTOUCH CT1720 Quick-Start Touch-Panel Controller

CUTOUCH CT1720 is an integration of a Touch panel, graphic LCD, and programmable embedded computer. Based on Comfile's CUBLOC CB290 PLC-on-a-chip, CUTOUCH CT1720 provides fast processing speed, 91 I/O ports, eight channels of 10-bit A/D, 6 x 16-bit PWM outputs, and 80 KB of Flash program memory so you can quickly develop HMI devices for industrial machines, factory temperature controllers, packing machines, robots, embedded control, and more.

Implementing a touch screen and controller can often add up to a lot of time and expense, but CUTOUCH CT1720 allows you to program working touch-buttons within the first few minutes. If you are thinking about developing a device that uses a touch screen, CUTOUCH CT1720 offers a very quick way to get to a finished solution.

CUTOUCH CT1720 is programmable in both Basic and Ladder Logic, allowing fast control, complex math, updateable touch-screen graphics, and fast data-communication protocols to be easily implemented. Ladder Logic offers real-time sequential processing and Basic supplies the number- crunching power. Both the real-time processing powers of a MODBUS PLC and the 32-bit floating point math, graphic capabilities, and communication powers of Basic are now available in one product.

CUTOUCH CT1720 has 82 I/O ports, and can be expanded with add-on boards to suit almost any situation (wireless, relay outputs, etc.). Using an optional XPORT Internet module, TCP or UDP packets can be monitored through the Internet from anywhere, allowing users to update or provide customer service for products located anywhere in the world.

With CUTOUCH CT1720, Basic can be used to draw graphics and print characters to the LCD and receive touch-screen input. Sensor signals enter through I/O or A/D lines, allowing you to turn relays on/off, output analog values, or send RS232 communication very easily compared with traditional non-Basic controllers.

CUTOUCH CT1720 has 28KB for data memory, RTC, and one of the two RS232 serial ports can be used for download and debug. An internal battery provides safe data backup. MODBUS support (Slave, ASCII) is also provided.

The CUTOUCH CT1720 Starter Kit is available now

from $362 from stock.

For further information, please contact:

Saelig Company, Inc.
1•888•7SAELIG Fax: 585•385•1768
Email: info@saelig.com
Website: www.saelig.com

MECHANICS

Get Your Ball Bearings!

Boca Bearings announces their new expanded range of Full Ceramic and Ceramic Hybrid ball bearings. Ceramic bearings are made of a highly manufactured ceramic, similar to the heat absorbing, super resilient tiles on the Space Shuttle. Ceramic is the perfect material for any application seeking to achieve higher RPMs, reduce overall weight, or for extremely harsh environments where high temperatures and corrosive substances are present.

Ceramic silicon nitride balls, for example, exhibit much greater hardness than steel balls resulting in at least 10 times greater ball life due to the ability to hold the surface finish longer. The ball has dramatically smoother surface properties than the best steel balls, resulting in less friction between the balls and bearing race surfaces. Thermal properties are also dramatically improved over steel balls, resulting in less heat build-up at high speeds. Ceramic has 35 percent less thermal expansion, 50 percent less thermal conductivity, are lighter weight, and are non-corrosive.

Similarly, the inner and outer races of anti-friction bearings often become frosted, fluted, or can get a corrugated pattern imprinted on them. These are not mechanical scars but are due to electromagnetic forces and can lead to bearing failure. They are usually found in modern systems that routinely feature pulse-modulated, adjustable-speed motors and inverters with high switching frequencies and short rise times. The best solution substitutes ceramic hybrid bearings for the more traditional, chrome steel counterparts to eliminate scarring and also to run cooler due to less micro-weld adhesion.

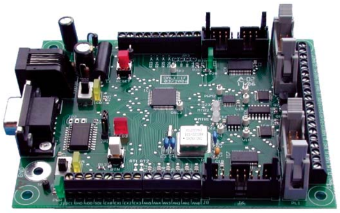
Suitable applications include cryopumps, medical devices, semiconductors, machine tools, turbine flow meters, food processing equipment, robotics, and optics. The Boca Bearing Company stocks a full range of ceramic balls, ceramic hybrid bearings, and full ceramic bearings. With over 2,500 different bearing sizes and well over two million bearings in stock, Boca Bearings offers a large stock of replacement bearings for all industrial and specialty applications.

For further information, please contact:

Boca Bearings
Tel: 800•332•3256
Email: clara@bocabearings.com
Website: www.bocabearings.com

MOTOR CONTROLLERS

New Closed-Loop Dual Motor Control System



Embedded Electronics, LLC of Philomath, OR has announced a new feature-rich Dual Motor Controller ("Dalf"). The board interfaces with standard motor drives expecting Signed Magnitude PWM control signals and provides both open and closed loop control of brushed PMDC motors.

Closed-loop features include robust PID and Trapezoidal Generator firmware to ensure smooth position and velocity control. Closed-loop feedback is via standard quadrature incremental encoders. Support for PID Motor Tuning to optimize system response is provided via data capture using the Step Response command. Open-loop control is supported by two R/C (standard 1.5 ms centered pulse) modes on three channels along with two analog voltage control (Pot) modes (two channels). Adjustable slew rate controls provide smooth velocity transitions for both open- and closed-loop operations.

Three separate serial command/monitor interfaces (Terminal Emulator, binary Application Programming Interface (API), and I^2C) support off-board communication and control of all open- and closed-loop features. The serial interfaces are functional in all operating modes, including the R/C and POT modes. A Windows GUI using the API is under development.

Motor and electronic protection is provided in hardware and firmware with support for current limiting using off-board, Hall-type, current sensors.

The board utilizes the PIC18F6722 microcontroller running at 40 MHz and supports additional code development using standard Microchip tools and the six-pin modular ICD programming connector. The firmware features — implemented with a mix of C language and PIC Assembler — are interrupt driven for efficiency. A parameter block in non-volatile memory provides storage for motor parameters, operating mode, and other power-up settings.

The C language source including the main loop and services requested by the interrupt handlers is provided. A library of functions, callable from C, provides easy access to all on-board devices from user written code. There is ample headroom for custom or extended applications with lots of unused memory (FLASH, RAM, and EEPROM) and processor cycles. Extensive I/O connections are provided including 32 GPIOs from I/O expanders, as well as digital, analog, and interrupt capable pins all routed to connectors for off-board use. A serial boot-loader is supported for in-application code upgrades without need for an ICD programmer.

Extensive documentation is available for download

including an Owner's Manual, a Getting Started Manual, and specifications for the serial command interfaces from the Embedded Controller website.

Available now, units may be purchased from the Robot Power website (**www.robotpower.com**). Price is $250 each in single quantities. Volume and reseller discounts available.

For further information, please contact:

Embedded Electronics LLC
Tel: **541•929•9553**
Email: **support@embeddedelectronics.net**
Website: **www.embeddedelectronics.net**

ROBOT KITS

Take Education Off-Road

Rogue Robotics introduces the new Rogue ATR ERS™ (ATR — All Terrain Robot, ERS — Educational Robotics System) robot kit. This system is the first of its kind for high school classrooms and hobbyists, providing robotics, electronics, and object-oriented programming in one system, while offering unparalleled all-terrain mobility.

Rogue ATR ERS features an eight-inch base with rubber tracks, Rogue's universal sensor mount system, dual DC gear motors, extra level capability for expansion, and a 1.1 amp dual H-bridge module, extra level capability for expansion, a 7.2V NiCad battery, and an OOBoard™ educational development board as its brain. The Rogue ATR ERS is made from the same laser cut, powder coated aluminum as the popular Rogue Blue robot base.

The Rogue ATR ERS is bundled with a curriculum text full of experiments, a parts kit, and a plastic storage box to house the fully assembled robot neatly in a classroom or under your workbench.

The feature-packed OOBoard, embedding the OOPIC® object-oriented processor, which can be programmed in C, Java™, or Basic syntaxes, powers the Rogue ATR ERS. The kit includes a CD-ROM that contains the programming editor for the OOBoard, as well as samples and curriculum materials.

The Rogue ATR ERS is "the SUV of Educational Robots," says Brett Hagman, Vice-President of Rogue Robotics. "No longer are small obstacles, uneven floors, or cables barriers for your robotics experiments."

The Rogue ATR ERS robot kit sells for US$324.95 and the OOBoard sells for US$119.

For further information, please contact:

Rogue Robotics
103 Sarah Ashbridge Ave.
Toronto, ONT M4L 3Y1
CANADA
416•707•3745 Fax: **416•238•7054**
Email: **info@roguerobotics.com**
Website: **www.roguerobotics.com**

EVENTS CALENDAR

Send updates, new listings, corrections, complaints, and suggestions to: **steve@ncc.com** or FAX **972-404-0269**

Know of any robot competitions I've missed? Is your local school or robot group planning a contest? Send an email to steve@ncc.com and tell me about it. Be sure to include the date and location of your contest. If you have a website with contest info, send along the URL as well, so we can tell everyone else about it

For last-minute updates and changes, you can always find the most recent version of the Robot Competition FAQ at Robots.net: **http://robots.net/rcfaq.html**

— R. Steven Rainwater

October

14 Robot-Liga
Kaiserlauter, Germany
Includes mini sumo, line search, labyrinth, master labyrinth, robot volley, and robot ball.
www.robotliga.de

17-20 Russian Olympiad of Robots
Moscow, Russia
A wide range of events for autonomous and remote-controlled robots including fire-fighting, line-following, cross-country racing, RoboCup soccer, vacuum cleaning, and combat.
http://intronics.bogorodsk.ru

20 Elevator:2010 Climber Competition
Las Cruces, NM
Autonomous climber robot must ascend a 60 meter scale model of a space elevator using power from a 10 kW Xenon search light at the base.
www.elevator2010.org/site/competition.html

27-29 Critter Crunch
Four Points Sheraton Hotel, Denver, CO
Held in conjunction with MileHiCon. See robot combat by the folks who invented robot combat competitions.
www.milehicon.org

November

18 DPRG RoboRama
The Science Place, Dallas, TX
Events include Quick-Trip, line-following, wall-following, T-Time, and Can-Can.
www.dprg.org/competitions

18-19 Eastern Canadian Robot Games
Ontario Science Centre, Ontario, Canada
Multiple events including fire-fighting robots, sumo, BEAM photovore, BEAM solaroller, a walker triathalon, and art robots.
www.robotgames.ca

24-25 Hawaii Underwater Robot Challenge
Seafloor Mapping Lab, University of Hawaii, Manoa, HI
ROVs built by university and high-school students compete in this event, which is part of the MATE (Marine Advanced Technology Education) series of contests.
www.mpcfaculty.net/jill_zande/HURC_contest.htm

24-26 All Japan MicroMouse Contest
Nagai City, Yamagata, Japan
Includes Micromouse, Micromouse Expert level, and Micro Clipper events.
www.robomedia.org/directory/jp/game/mm_japan.html

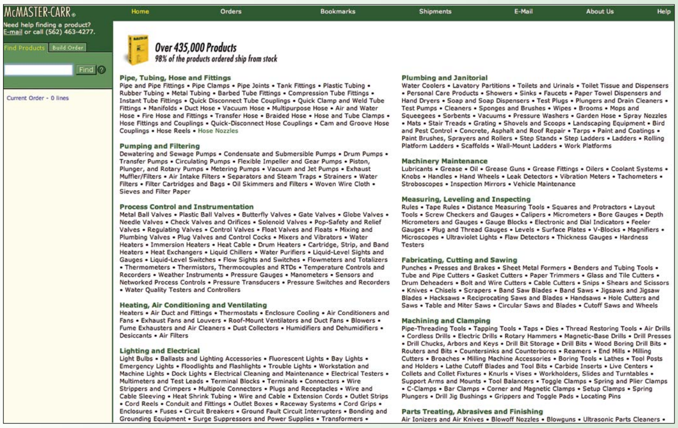
December

1-2 Texas BEST Competition
Moody Coliseum, SMU, Dallas, TX
Students and corporate sponsors build robots from standardized kits and compete in a challenge that changes each year.
www.texasbest.org

2 LVBots Challenge
Advanced Technologies Academy High School, Las Vegas, NV
Line-following, line maze solving, and mini sumo; all for autonomous robots.
www.lvbots.org

8-9 South's BEST Competition
Beard-Eaves Memorial Coliseum, Auburn University, Auburn, AL
Regional BEST teams from multiple states compete in this regional championship.
www.southsbest.org

TidBOTs



Get ready for a real treat — the McMaster-Carr online catalog is a great source for all sorts of stuff. Although it is a bit wordy.

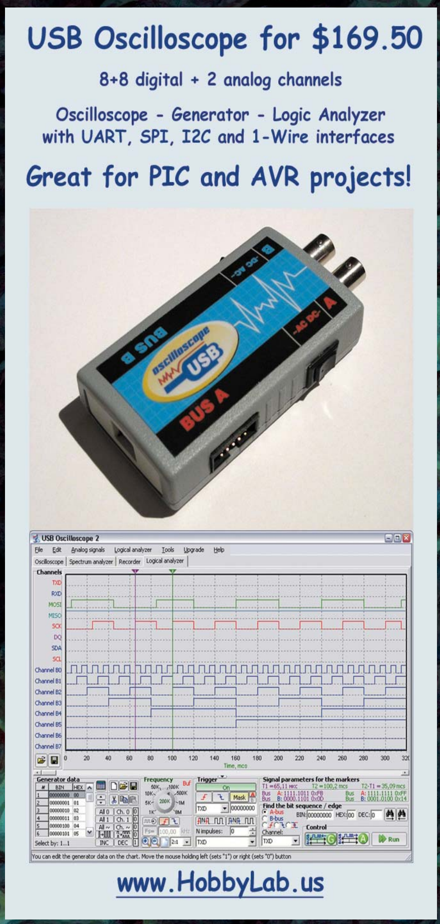
A Monster Catalog

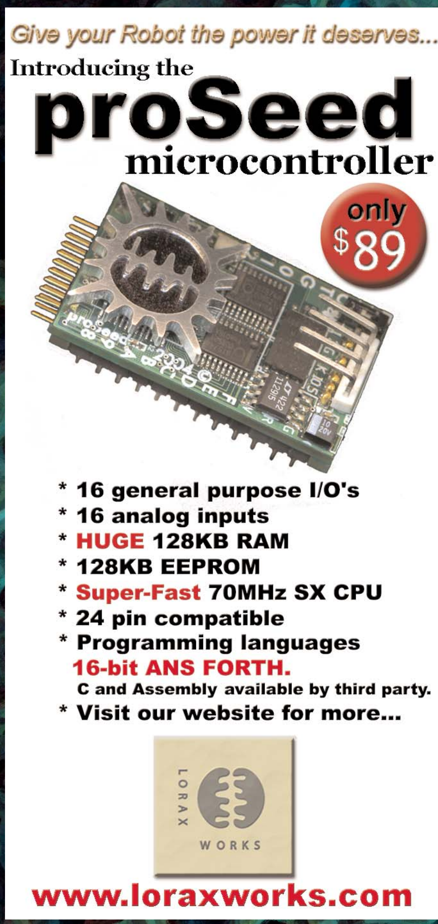
Are you looking for a bellows? How about some welding equipment? Or, an air cylinder, roller chain sprockets, woven wire cloth, and piston plunger? These items and over 435,000 others can be found "inside" the McMaster-Carr website online catalog (**www.mcmaster.com**). Forget about those conventional paper catalogs that waste your bookshelf space! The McMaster-Carr digital catalog is quick, easy, and powerful. Granted, it does look daunting in its opening screen, but there is a search feature if you need to use it. For me, half the fun is drilling down through the product lists looking for a needed product. Naturally, my final order is littered with a lot of other "stuff" that was found through pure serendipity. Ya gotta love that serendipity!

Roomba Humor

I came across a podcast from "The Onion" website (**www.theonion.com/content/node/51389**) that might give you stuffy robot experimenters a chuckle. Apparently a Roomba has been secretly collecting evidence for use against its human captor. I only wish that the recorded Roomba "interview" had featured the robotic floor vacuum system's warning "chime" rather than the sucking noise from a conventional vac. **SV**

Robotics Showcase





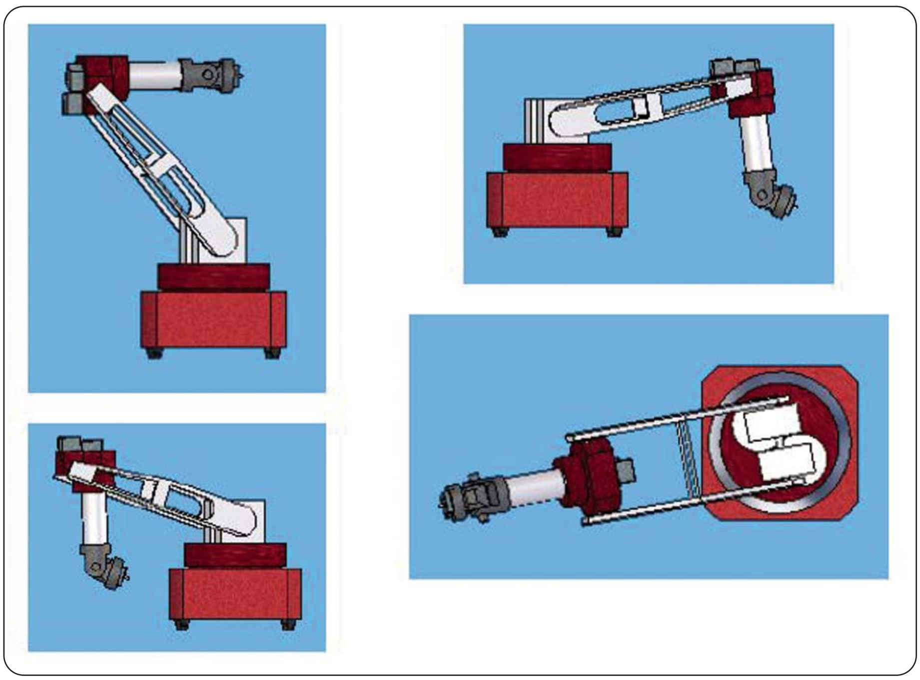
MENAGERIE

Builder's Name: Francisco C. Oliver Rivera
Significant Robot Building Milestone: Select the motor power for the different arm positions. Inverse kinematics.
Robot's Name: None yet.
Robot's Reach: 800 mm (31.7")
Robot's Weight: 20 Kg (44 lb.)
Significant Robot Living Milestone: Successful test with only nine months for design and building. I get an "A" in my Engineering Deg.

Additional info:

Reach	600 mm (23.7")
Payload	0.5 Kg (2.7 lb.)
Degrees of freedom	6
Speed	0 to 0.2 m/s (0 to 0.008"/s)
Accuracy	±3 mm (0.12")
Motors	Six stepper motors (200 steps per turn)
Brain	SAB80C537
Stepper motor controller	L297
Dual full-bridge driver	L298

My email kicoymaria@hotmail.com



COMBAT ZONE

Featured This Month

PARTICIPATION

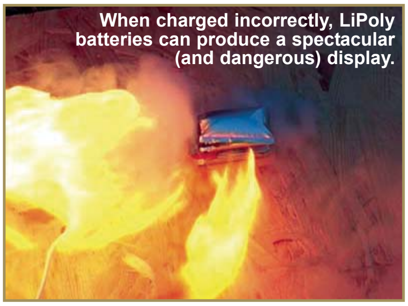
Battery Safety 101

● by Michael Rogers

Lithium manganese, nickel-cadmium, nickel metal hydride, sealed lead acid; like food to a human body, batteries are the energy source that combat robots are powered by. Batteries typically discharge stored energy to power a load, such as a motor. However, in certain situations — such as a dead short where the battery is essentially discharged instantaneously — a battery can ignite and release toxic gases. Needless to say, there's no need to tremble every time you see a battery because of its explosive potential. In this article, I will cover the finer points of proper battery safety in order to keep you safe and your batteries performing optimally.

The three main safety topics for batteries concern: wiring, charging, and storing. Although the utmost care should always be taken when dealing with all types of batteries, I can't stress how important it is to use extra caution when using particularly energy-dense cells — such as lithium polymer batteries — which are notoriously fragile and can be highly explosive.

First, wiring up your robot in a safe and organized manner will not only help prevent shorts, but it will also make general robot maintenance easier; this includes using proper gauge wire for your application and using two distinct wire colors to clearly distinguish your positive and negative connections. In addition, make sure that all your connections are covered with either shrink wrap or electrical tape. After you have completed your wiring, double and triple



When charged incorrectly, LiPoly batteries can produce a spectacular (and dangerous) display.

Some teams have custom charger setups to reduce workbench clutter and setup time at events. Photo courtesy of Killerbotics.

A safe charging setup for LiPoly batteries, using a glass container on a plastic table. Photo courtesy of Team Hammer Bros.

These packs show the damage caused by excessive discharge current. Photo courtesy of Team Mad Cow.

check all your connections to check for any possible short circuits.

Once you're finished wiring and have carefully checked everything over a few times, you're ready to charge your batteries. It's best for your battery's life span to trickle charge your batteries over a long period of time at a lower amperage. As a rule-of-thumb, you can charge your batteries at the amperage that your batteries are rated to discharge. For example, you can charge a 2,000 mAh (2 Ah) pack at 2.0 amperes. I would highly recommend that you use a fireproof LiPoly charging bag which can contain a full-on LiPoly pack meltdown and protect your surroundings from nasty fires.

You might think of battery charging as something where you "set it and forget it," however, it's very important that you always keep a close eye on your batteries and make sure you know when they peak. Believe it or not, you can over-charge your battery packs which can lead to a release of hydrogen gas.

Once you've (hopefully) had a successful foray into the competition process and are perhaps ready to store your batteries, make sure to place them in a cool, dry location. Sealed lead acid batteries should be fully charged when stored, where nickel and lithium based cells should be stored with a 40%-50% charge.

Ultimately, this article is only a short blurb on battery safety that should provide sufficient information to allow proper basic battery use, however, battery distributors such as **www.robotpower.com** can provide you with more in-depth information on battery safety. SV

Pit Repairs

● by Wendy Maxham

Pit repair begins in the design process. Whether you design your robot on the computer or by laying out the parts in a tape outline, think about which parts will need to be accessed. Batteries are a prime example: If you plan to remove your batteries between fights (best bet if you use Nicad or NiHM, not as necessary for SLAs), then make sure they aren't buried inside the bot. Even if you plan to charge your batteries without removing them from the bot, remember that batteries don't like to be charged while hot. You can use a fan to cool the batteries, but if they are buried deep inside your bot, the fan might not provide enough cooling to allow a full recharge.

Hardware is an important design decision too. Minimizing the types and sizes of bolts and screws in your bot will also minimize the number of tools and replacement parts you need in the pits.

Once you leave the arena, go to your pit table and get started with your maintenance or repairs. All competitions guarantee a minimum time period between fights, usually 30-45 minutes. In the early rounds of a competition, you could have all day to make repairs, but you should still get started immediately. When you open up your robot after a fight, you might find other maintenance issues you need to take care of: loose nuts, broken speed controller fans, and loose wiring.

If you know you have a lot of repairs to make, have someone check the fight schedule to see when your next match is supposed to be. If you are unsure if you can make all the necessary repairs in this time frame, talk to the fight scheduler to see if your fight can be postponed. Understand that event timing might not allow for a postponement. In that case, you have two options: forfeit you next fight or dig in and make what repairs you can.

After you've made the decision to make the repairs, prioritize what needs to be done.

● **#1 Batteries** — Get them charging — or better yet, have a backup set

PHOTO 1. Devil's Plunger as it's *supposed* to work.

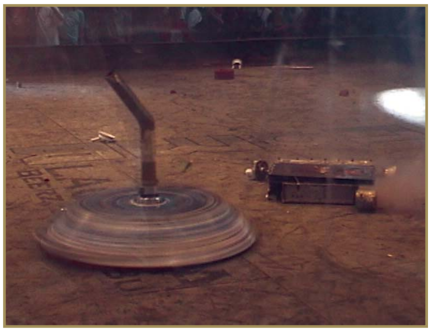


PHOTO 2. Megabyte dismantles Devil's Plunger.

already charged. Don't forget the transmitter and receiver batteries (if you don't have a battery elimination circuit or BEC). We almost lost a fight once when the transmitter started beeping a low battery warning.

● **#2 Drive Train** — Mobility is essential. Do what it takes to make your bot mobile again. Start with the basics of getting the wheels turning (replacing motors, ESCs, broken chains/belt, blown out bearings). Then you can determine if you want to move on to #3 before repairing things like slightly bent axles. You would replace a bent axle if you also have to replace a blown bearing or broken sprocket, but consider leaving a bent axle if it still turns okay and you still need to make other critical repairs. If you had to rewire the drive train during repairs, get the frequency clip (if possible) in order to test the drive before your next fight.

PHOTO 3. The winner and the pile of parts. Photo courtesy of Michael Mauldin, Team Toad.

● **#3 Weapon** — It's usually best to go into a match with your weapon working, but one thing to consider: Will your repairs likely hold up in battle? If not, you might think about removing the weapon before a fight or go into the fight without it working. When you start a fight with your weapon working then it stops working, the judges' view this as damage to the bot caused by the fight. Since most competitions award more points for damage, putting a marginal weapon in the arena can cost you points in a fight.

● **#4 Cosmetics** — Check the schedule again to see when your next fight is (if the scheduler isn't already standing over your shoulder calling you into line). If you still have time, you can take your bot to the designated grinding area and grind/sand down sharp spots, straighten bent armor, sharpen wedges, etc.

If you need help with repairs, ask around the pits. Many builders (sometimes even your next opponent) are more than happy to help get a bot ready to fight. That being said, it's also okay to refuse help.

A Tale From the Pits

At RoboGames 2006, we decided to fight our MW Devil's Plunger as a Heavyweight since the new flamethrower wedge put DP at about 140 pounds. DP's first two fights were against nasty spinning robots, so we used a spinner defense wedge. In the first fight, the spinner defense wedge held up well against horizontal bar spinner Last Rites. Round two pitted DP against full-body spinner Megabyte. Megabyte kicked Devil's Plunger all around the arena for two minutes, ripping off the spinner defense wedge, as well as all eight wheels, including one that was still attached to the corner of DP (see Photo 2). From the flashes of flame and smoke pouring out of DP, we knew there were some internal issues to deal with, too.

When we got back to the pits, DP looked pretty hopeless (see Photo 3). The front right corner was missing (which included a wheel mount and one side of the wedge mount), one motor was seized up, two speed controllers were fried, several sprockets broken, bearings blown out, remaining axles bent or damaged, radio box shattered, and antenna wire cut. We figured DP was out of the competition and permanently retired!

After we got back from fighting our other robot Sewer Snake, we reassessed Devil's Plunger and thought maybe it could come back for one more fight with the flamethrower wedge. We checked with the fight scheduler Marc and he said DP's next fight would have to take place that evening. Worst case scenario — we had about three hours to get DP running again.

First task was to get all of the broken parts out of the robot and make sure we had replacements. I started stripping DP down while Matt assembled the part necessary

PHOTO 4. Repaired and ready to roll. Photo courtesy of Felipe Scofano, Team Riobotz.

PHOTO 5. Devil's Plunger vs. Full Smash.

for the rebuild. Working quickly but methodically over the next few hours, we replaced the motor, speed controllers, bearings, sprockets axles, and chains. Since we replaced two speed controllers, we had to test the drive to make sure the motors were wired correctly. Amazingly enough, DP fired up and the wheels actually turned the right direction!

With the drive train back in working order, we turned our attention to the weapon. Usually DP's wedge mounts between the two front shoulders. With one shoulder gone, Matt had to get creative. He used an old Sewer Snake weapon shaft that had large washers and nuts on the end to capture the wedge on the remaining shoulder. Then using nylon truck tie-downs, Matt strapped the free-floating side of the shaft to the bot. This helped keep the wedge from pulling away from the bot while maneuvering around the arena.

About 3-1/2 hours after we started repairs, DP was ready to fight (see Photo 4). Cosmetically, there wasn't much we could do for Devil's Plunger. Bright red duct tape covered the hole in the right front where one wheel used to be. One thing we hadn't considered during the repairs became evident as Matt went through the start-up sequence before the fight.

Matt stuck the Allen wrench through the receiver switch hole, forgetting the receiver switch wasn't in the same place since Megabyte destroyed the receiver box. Matt had to work his way through the duct tape to find the receiver switch. Not only did DP make it back into the arena that night, he won the fight against Full Smash (see Photo 5)!

One member of the Brazilian team RioBotz said she felt like crying when Devil's Plunger was dismantled by Megabyte. I, on the other hand, was much closer to tears as we left the arena after the Full Smash fight to a standing ovation from the audience and builders. It's difficult to beat the feeling of putting a pile of pieces back together into a working machine against the odds. Part of the challenge in robot combat is getting a bot to an event, the other part is keeping it going at the event. That's what this sport is all about — build, fight, repair, repeat. SV

Cherry Box

Watch out for sharp spots on your robot that may not have been there before the fight. Even though you're familiar with your bot and its normal sharp points, new sharp spots are created during combat.

SHARPENING THE SWORD
Evolution of a Combat Robot

● by Russ Barrow

The sharpest sword is the sword sharpened most often. Combat robotics is an excellent venue for demonstrating building design and execution. The sheer competitive environment and consequence of competing enforces determined dedication.

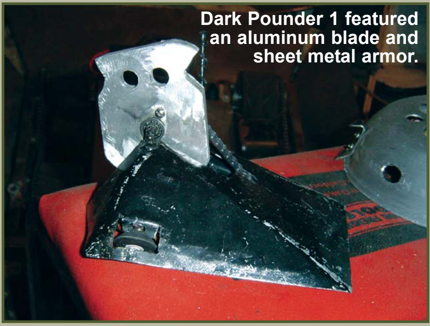
I have been building and competing with combat robots for almost five years. I have built many robots and tried various designs. Some worked, some died (sometimes in spectacular fashion). The ones that worked provided the positive reinforcement to continue evolving the design. The following is the build evolution of one of my first combat robots in the Ant or one-pound combat class. The Robot is fittingly called Dark Pounder.

For the work I have done with Dark Pounder, not only was the design successful, the design was able to be improved to overcome my often poor execution. In 2005, Dark Pounder was able to achieve the highest honor or a #1 ranking in the Ant class of Robotic combat. The bot was also elected to the Robot Combat Hall of Fame with an Honorable Mention.

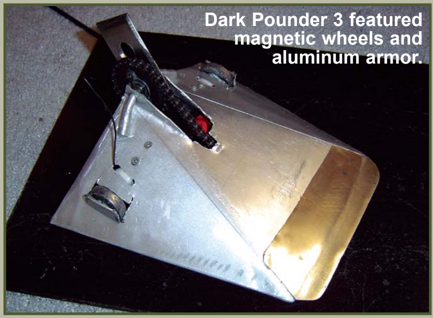
Dark Pounder Version 1

Dark Pounder began life in January of 2003. The design is classified as a vertical spinner with the weapon disk or blade spinning upwards, facing the opponent. Pounder was a very different looking bot; I refer to it as a very fortunate mistake. The step side of the wedge was meant for attack, but in its first match, I could not get a bite on the competitor. So for the next few fights of the event, I reversed the direction of the weapon and drove it backwards with the low wedge getting under the other bots and flipping them.

Version 1 was made of thin galvanized sheet metal (found in any hardware store) and giant block of 1/2" aluminum for a weapon and a chain drive. It was made mostly of copier machine parts. The copier


Dark Pounder 1 featured an aluminum blade and sheet metal armor.

Dark Pounder 2 in action during a SWARC event.


Dark Pounder 3 featured magnetic wheels and aluminum armor.


Version 4 upgraded to titanium armor.

donated bearings, shafts, gears, and chains. I used two RC servo controller boards to control the two 16 mm drive gear motors. I ran six NiCd cells to a Speed 280BB RC airplane motor.

The bot was a winner right out of the gate, compiling an 11-0 record, but the competition was closing in. This bot was rebuilt after two competitions because it was very slow and non-invertible. It was parted out in June 2003 — after a mere six months of life.

Dark Pounder Version 2

The aluminum block on version 1 was limited to about 6,000 RPM due to poor aerodynamics, so a blade would make more sense. In addition, a thinner spinning weapon would concentrate the weapon energy and add a cutting effect. I chose a 5" length of 1/8" stainless steel. The blade appeared to spin a bit faster, but the stainless steel was substantially heavier than the aluminum block. Since the blade was so heavy, I used a super thin stainless spring steel I found at a local surplus store for armor. Who needs strong armor when you have an aggressive weapon ...

Well, another competitor arrived with rare Earth ring magnets on all four wheels. His robot attached itself to the metal arena floor with over 16 pounds of effective weight. He came at me and I kept hitting him but giving up ground the whole time. He eventually pinned me up against the wall and began crushing the bot. Dark Pounder still managed to go 4-2 in this July 2003 event, but my eyes had been opened to ring magnets for drive wheels. Perhaps a new design using magnets was in order.

Dark Pounder Version 3

Other competitors had used magnets on the steel arena floor, but unknown to me was how powerful magnets had gotten. The magnetic force-to-weight ratio was dramatically better than common speaker magnets. So, how do you defeat another robot that is basically glued to the floor? Well, a very sloped wedge would work nicely.

To make a gradually sloping wedge would require some changes, the biggest one being the size of the blade. I hated giving up blade length, since it would seem a bigger blade would deliver more energy. However, I can spin a smaller blade faster, and even with less mass, deliver a more energetic hit (Energy = 1/2Mass x Velocity^2). I targeted to spin the blade at about 12,000 RPM. The shell was made from one piece of aluminum, vise formed, to offset the additional weight of the magnet wheels. I used paper CAD to work out the angles.

This design lasted for almost a year, and included improvements to the weapon drive and batteries. The chain weapon drive was simply too slow, so I used a small rubber O-ring and some small belt pulleys from the copier machine. I was able to get a 4.5:1 gearing, which improved the speed of the blade and also reduced the weapon motor current. The rise of lithium polymer batteries provided an opportunity to provide a higher voltage and lower weight than the NiCad batteries, while maintaining the same capacity.

Dark Pounder was virtually invulnerable on steel, and one of many reasons arenas moved to non-ferrous floors (such as wood, stainless steel, and aluminum). This version of Pounder went 20-3 from October 2003 to September 2004. Two losses on a non-wood arena floor eventually revealed that the aluminum frame was not strong enough for the vastly improved competition. Maybe titanium would work better ...

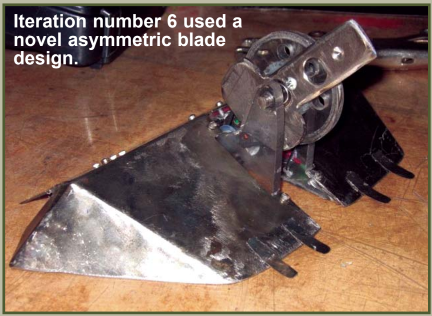
Dark Pounder Version 4

Some lessons are more difficult to learn; this was the case for Pounder version 4. Pounder 3 was a successful bot, but after a year of minor improvements, a rules change had impacted the design. Unfortunately, I did not fix the problem but tried to solve the effect. Without magnets, other wedges managed to get under it. A stronger frame just made the bot more survivable, but did not solve the problem that I had to get under my opponent to deliver the weapon. Version 4 was nearly identical to version 3, but the AL skin was replaced with titanium.

In competition, the shell certainly could take some punishment, but delivering the weapon became more difficult. Version 4 failed because other powerful spinners need only one hit on the front wedge to throw me across the arena, as happened at the Nationals 2004 competition from two formidable vertical spinners.

Version 4 went 2-2 in this one

Dark Pounder 5 was a major change of concept and fabrication techniques.



Iteration number 6 used a novel asymmetric blade design.

The latest Dark Pounder — number 7 — waits for trial by combat.

competition. The writing was on the wall — I must find a way to deliver the weapon without depending on the wedge. This version did teach me the toughness of Ti, and how to work with it. Some heat and patience can make a very solid shell.

Dark Pounder Version 5

Version 4 had proven that a new shape was required, but the ruggedness of titanium and impressive results people were getting with carbon fiber required more materials investigation. Because titanium was both stronger and lighter than steel, I ditched the stainless steel blade and went with a larger 5" Ti blade.

To better deliver this weapon, I needed to push out the blade from the robot, so the previous shell design would not be possible. This bot would require a frame. Polycarbonate offered a perfect solution. This material can also be easily machined or cut into any shape I needed. Next, I used three carbon fiber RC airplane tubes to connect the polycarbonate and also act as the mounting shafts for the drive and weapon motors. Cover it in TI, and it would be a very tough bot.

Now, how could I guarantee to get under wedge-based robots? The key would be focusing the pressure (or force). A point produces more pressure per area than a line by concentrating the force. So, by using some titanium plates cut to a point, I could get under the flat line of a common wedge.

This worked very well. Version 5 managed a very respectable record of 16-4, and was around for almost a year. In the end, the lack of a wedge surface on the sides of the bot made it susceptible to horizontal spinner hits and wedges.

Dark Pounder Version 6

Version 6 was another dramatic change from the previous design. Instead of trying to fix the few problems that existed with a very successful design, I decided to start almost from scratch again. This bot would have a wedge surface for all surfaces, so a strong defense was in place. I had also decided to try to implement an asymmetric blade design.

The one problem with spinning a bar or disk at a very high rate of speed is that you will have difficulties delivering the energy from the blade simply due to the inability to get a good bite on the opponent. The bite is determined by how much contact the blade or disk can make, which is a direct result of the differential speed of the two bots, and the distance the blade or disk teeth travel in that time. So, at high RPM, a symmetric blade or disk only has half the time of the blade or disk rotation to put the opponent into the weapon. With an asymmetric weapon like the one on Version 6, you can get an entire blade rotation and therefore double the time to put the opponent into the weapon.

Construction of Version 6 was nearly identical to Version 5. Unfortunately, I tried placing the wedge in front of the weapon similar to the earlier Pounders. Once again, the design won or lost based on if the wedge could get under the opponent.

Version 6 managed to go 6-4, existing for only five months. For the first time, I did not immediately start building another version. Pounder would need a complete rethink, and it took almost five months before an idea would surface. What would over three years of development produce?

Dark Pounder Version 7

For five months, I worked on other bots, tried a few new ideas in a few different directions. At first, I thought I would rebuild Version 5 with a wedge surface on the sides. A new design was in order that required the bot to be more compact. A lower center of gravity would make the bot more stable by reducing the gyroscopic force inherent in a fast vertical spinning object.

A smaller blade would need to be designed. I liked the asymmetric blade idea, but I needed to move the moment of inertia (center of the spinning mass) further out than the earlier design blades. I liked the hook shape blade since it would tend to bite with a grabbing type of force. I would also want to sharpen the blade surfaces to an edge to improve aerodynamics and the cutting potential of the blade. I made the blade from a piece of 1/8" chromoly steel that I cut, balanced, and heat treated.

On another bot called Dark Micro 44, I had a rounded wedge that proved to be very strong, as well as capable of deflecting energy. So after mocking up the idea using paper, I cut a 10" semi-circle of .03" titanium and then spent about an

hour bending it to the desired shape.

The frame would continue to be made of 1/4" polycarbonate, but I would use a titanium base-plate reinforced with a carbon fiber rod for rigidity. This would protect the bot top to bottom, and act as a mounting point for the drive motors.

To make this bot destructive, I designed the weapon blade to spin at 22,000 RPM. The Speed 280BB RC airplane motor I have been running spins at 66,000 RPM off of four Li-Poly batteries, and is geared down 3:1 to the weapon. The motor and batteries do get warm from extended running, but I hope most matches will end quickly. Will this new design work? Check with your local robotic combat clubs/events and find out.

Amazing what can be done with only a pound of weight and a little imagination (not to mention a considerable amount of time). SV

EVENTS

RESULTS — July 11 - August 14

WBX-3 was hosted by War-Bots Xtreme and was held July 22nd in Saskatoon, Saskatchewan. Fifty-six bots were registered in all classes from Antweights to Heavyweights. Results are as follows:

- *Ants* — 1st: "Glitch 2," pusher, Chaos Robotics; 2nd: "Buggy 2," wedge, X-Bots; 3rd: "Hoser'd Reloaded," spinner, FingerTech.

- *1 KG* — 1st: "Roadbug," wedge, Chaos Robotics; 2nd: "Hell's Angle," wedge, Tru Pride; 3rd: "Bot and Paid For," wedge, Humbot.

- *Beetles* — 1st: "Flippenstein," pneumatic flipper, FingerTech; 2nd: "Creepy Crawler," wedge, X-Bots; 3rd: "Limblifter," lifter, GuavaMoment.

- *Mantis* — 1st: "G.I.R.," drum, Chaos Robotics; 2nd: "Blenderhead," full body spinner, Inner Logic; 3rd: "Banshee," ICE spinner, Chaos Robotics.

- *Hobbyweight* — 1st: "Bubba," pneumatic lifter, X-Bots; 2nd: "Dexahedron," drum, GuavaMoment; 3rd: "Ranch Tooth," hammer, Rumble Robotics.

- *Featherweights* — 1st: "Nearly Normal," wedge, LNW; 2nd: "Mean Machine," spinner, X-Bots; 3rd: "Slapped Together Wedge," wedge, Geek.

- *Lightweights* — 1st: "Agent 7," wedge, X-Bots; 2nd: "Fly Speck II," wedge, Maggot; 3rd: "MDC," spinner, Inner Logic.

- *Middleweights* — 1st: "Botfly," pneumatic flipper, Maggot; 2nd: "Speed Bump XL," pneumatic lifter, X-Bots; 3rd: "Maddgoth MK2," spike, Crash.

- *Heavyweights* — 1st: "LNW," spinner, LNW; 2nd: "Amata," drum, X-Bots; 3rd: "The Defyer," pneumatic flipper, Syberon.

Saturday Night Fights was hosted by Team Think Tank and was held July 22nd in Pasadena, CA. Twenty-three bots were registered in the Fairy, Ant, and Beetleweight classes. Results are as follows:

- *Fairyweight* — 1st: "Crisp," flamethrower wedge, Offbeat Robotics; 2nd: "Ugly Duckling," lifter, Slayer.

- *Antweights* — 1st: "Baby Blaster," spinner, Ghetto Logic Robotics; 2nd: "Rick James," spinner, Fatcats; 3rd: "Ducbot," lifter, Slayer.

- *Beetleweight* — 1st "Unknown Avenger," flipper, Ice; 2nd: "Roboslayer," wedge, Slayer; 3rd: "Bite Me," box, Slayer.

Bring Your Own Bots III was hosted by Team Cerberus and was held August 12th in Chuluota, FL. About 15 bots were registered in the UK Ant, Ant, and Beetleweight classes. This was an informal "front yard" event and included some great food and fellowship. Results are as follows:

- *UK Ants* — 1st: "Electric Eye," lifter, Cerberus; 2nd: "P150," spinner, Overvolted Robots.

- *Antweights* — 1st: "Serphiroth," spinner, Cerberus; 2nd: "Ultimate Ultimatum," spinner, Overvolted Robots.

- *Beetleweight* — 1st: "Hungry Hungry Hippo," Lab Rat Revolt; 2nd: "Plagiarist," spinner, BotWorks.

Robot Fighting League 2006 Nationals were held August 12 in Minneapolis, MN. Presented by the Midwest Robotics League, this is the top event in the RFL's fighting season. A more detailed article separate from the Combat Zone is included in this issue. Results are as follows:

- *Antweights* — 1st: "UnderWhere?!," spinner, Hazardous Robotics; 2nd:

"Pop Quiz," spinner, Test Bot; 3rd: "Anti," spinner, 564 Robotics.

• *Beetleweights* — 1st: "Itsa?," spinner, Bad Bot; 2nd: "Nuclear Kitten," spinner, Test Bot.

• *Hobbyweights* — 1st: "Cheapshot 3.0," wedge, Rolling Thunder; 2nd: "Surgical Strike," spinner, Rolling Thunder.

• *Featherweights* — 1st: "Xhilarating ImpaX," wedge, Rolling Thunder 2nd: "Killabyte," full body spinner, Robotic Death Company.

• *Lightweights* — 1st "Son of Whacky Compass," spinner, Hawg; 2nd: "Goosfraba," flaming wedge, Killerbotics; 3rd: "Death By Monkeys," wedge, Death By Monkeys.

• *Middleweights* — 1st: "Ice Cube," plow, Toad; 2nd: "Lunatic," spinner, Booyah; 3rd: "Lionheart," wedge, Toad.

• *Heavyweights* — 1st: "Shrederator," full body spinner, Logicom; 2nd: "Eugene," spinner, Moon; 3rd: "Ty," plow, Bobbing For French Fries.

• *Superheavyweights* — 1st: "Psychotic Reaction," spinner, kontrolled kaos; 2nd: "Star Hawk," spinner, Moon; 3rd: "The Wall," Moon. SV

TECHNICAL KNOWLEDGE

Gyro Usage in Combat Robotics

● by Eric Scott

In combat robotics, we rely on many industries to produce motors, power transmissions, and structural frameworks for our robots. For control of the robot, we generally rely upon the radio-controlled hobby industry. Understanding the application that the device was originally designed for can help us to better understand how best to exploit it for our own purposes. The focus of this article will be the modern radio control helicopter gyroscope, and how it can be used to improve our combat robots.

What Gyros Do

To start out with, let's take a look at what the gyro is used for in the modern day model helicopter. A standard single main rotor helicopter has a big problem: The torque effect generated by the main rotor wants to spin the helicopter around the main shaft. In order to counter this, the tail rotor is employed to produce an opposing thrust. The amount of thrust is varied on the tail rotor to either intentionally rotate the helicopter around the main shaft, or to correct for changes in rotor torque and to keep the tail stationary.

This is where the gyro comes into play. Everything from small wind gusts, to changing blade pitch, to specific aerobatic maneuvers requires very small, precise corrections in tail rotor thrust. Managing the other controls of the helicopter is difficult enough, so we rely on a gyroscope to help us keep the tail where we want it, and yet allow it to move when we want that, as well.

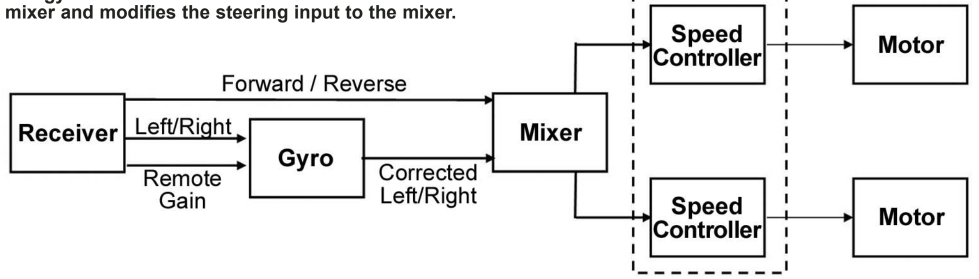
In a model helicopter, the gyro is used on the rudder channel, so as to sense changes in the rotation of the helicopter about the main axis and correct them. In a robot, we would use the gyro to sense changes from the commanded position on the turning axis, and correct for them. What would we need to correct for in a standard tank drive style robot, you ask? The answers are many: Changing arena conditions that affect traction on one side of the robot; binding drive train due to poor construction or damage; or unequally matched motors.

An omni-directional robot presents its own set of directional challenges. There are many things that can cause the robot not to follow the commanded direction. A good driver can somewhat correct for these, but why devote the brain cycles?

Rate and Heading Hold Gyros

Modern day gyros come in two flavors, only one of which we will really be interested in. These are known as "Rate" gyro and "Heading Hold." Rate gyros are the simpler

The gyro is wired between the receiver and the onboard mixer and modifies the steering input to the mixer.



The Futaba 401 features adjustments for delay and rate limiting. Photo courtesy of Tower Hobbies.

type. Imagine that you were to receive a one second pulse of wind from the side. The rate gyro would then apply one second of correction to bring the tail back around. This method is not all that precise, as it does not really take into account subtle drifts or variances in force applied. It is still better than nothing.

Heading Hold gyros memorize the last commanded angular position, and continue to apply corrective action until it gets back there. Futaba® refers to its Heading Hold technology as "AVCS," both mean one and the same thing. Heading Hold is the type we will be interested in.

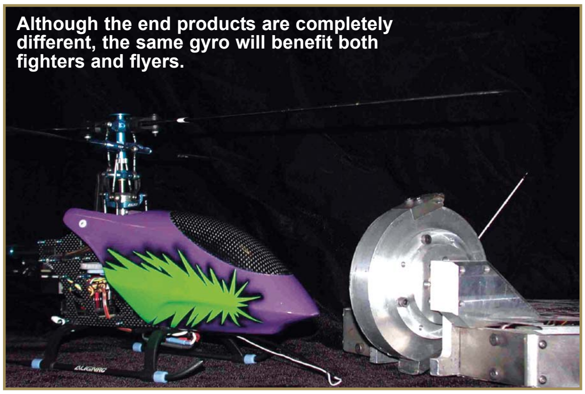
Applications for Robotics

So, what do you look for in a gyro for robot use? We've previously said that the gyro should be of the Heading Hold or AVCS type. A Rate type will only partially help us to correct unwanted motion, and is not worth considering in this application. Remote gain is a must in my opinion, not only for ease of adjustment, but it is essential for safety.

A robot with a gyro will try to correct itself back to its commanded position. Any force attempting to change the robot's direction will be "corrected" for, regardless of its origin. If a human were to pick up the robot and attempt to twist it, the gyro would send signals to correct this, causing wheels to spin, and a possibly dangerous situation could arise. With remote gain, one can remotely disable the gyro, making the robot safe to approach.

A very good all-around gyro for robot use is the Futaba® 401. It is an AVCS or Heading Hold type unit, with remote gain. It has proven fairly rugged for use in both combat robots and helicopter crashes and is relatively small and lightweight. All setup examples will be based on this gyro, although the principles involved will apply to any Heading Hold gyro. As always, please consult the manufacturer's instructions for use and installation.

In order to use the gyro in a robot, it must be allowed to correct changes in the yaw of the robot. For those of you used to mixing the two speed controllers in a tank style robot onto one stick in the radio, you will have to make a few changes. We need to do the mixing onboard the robot, so that the gyro has access to the "pure" turn channel.



Although the end products are completely different, the same gyro will benefit both fighters and flyers.

There are several onboard tank style mixers available. Connect the gyro so that it is plugged between the mixer and the turn channel of the radio. The gyro can be mounted anywhere in the robot provided that it is in the same plane as the motion that you wish to correct.

Do not place the gyro on its side. Placing it upside down is fine, as it still remains in the same plane. Use the mounting method the manufacturer recommends to secure the gyro, generally a piece of special double-sided tape. This tape reduces transmission of vibration from the bot to the gyro.

Testing

Now that you have the gyro installed in the bot, it's time to do some testing. Above all things — BE CAREFUL. You are introducing something into the control system which has the ability to make the robot do things you have not commanded if not set up properly. Before you do anything else, set up the remote gain function of the gyro, and assign it to a switch on the radio. Set it up such that you have zero gain in one position effectively turning off the gyro, and a small amount of heading hold gain in the other position. You should be able to check that the gyro is in heading hold mode by applying power only to the radio system, not the drive system. Take note that a Heading Hold gyro will take a couple of seconds to initialize during which the gyro must remain still. Do not move the robot during this time.

Limit Setting

Many gyros have a "limit" setting — a source of much confusion — as it is not a function we would often use. Its purpose is to limit the travel of a servo attached to the gyro. In the case of a helicopter, it would be used to prevent the servo binding at the ends of the linkage. Were you to want to do so, you could use this function to limit the amount of power commanded to the motors on the spin function of the robot. This setting should probably be left at 100. If you notice that you are not getting full

power during a spin, perhaps by observing the speed controls, you might increase this limit. In general, this control is of limited use to the robot builder.

The first rather critical step in setting up is to set the direction of the gyro. This is different from the direction that the stick causes the robot to move. This is set with a small switch on the gyro. Due to the variety of mounting options, it is impossible for me to tell you what this switch should be set at. If it is set incorrectly, when you spin the robot, the gyro will apply correction, but in the reverse direction. This will increase the rate of the spin, which will apply more reversed correction creating a feedback loop. This is why you MUST set the remote gain function up first, so that you can stop the robot in the event of a "death spin."

The direction is set correctly when the robot does not spin in this fashion. Perform all tests of this sort with the robot physically isolated from you by a curb or other barrier between you. A robot in this spin is very dangerous, and your reaction time to shut down the gyro gain may not be sufficient to prevent injury.

Gyro-Gain

Next, you will set the gain of the gyro. With a remote gain gyro, this will be adjusted in the radio itself. Futaba makes it easy if you use their gyro with their radio, as the radio has an easy setup screen to set the gain. If you are using different brands, consult your documentation (when all else fails, read the manual). You should generally raise the gain as far as you can until the robot shakes slightly or "hunts" for its position. When you get to that point, back off the gain slightly. You want the gain to be set as high as you can without shaking.

The Futaba 401 has an adjustment for "delay." This function is basically a dampening of the feedback loop. The delay refers to a delay in applying correction of the gyro in order to compensate for a heavy mass (such as a robot) that needs to be moved. When driving around, spin the robot and stop sharply. If you get a little wag when stopping, adjust the delay upwards a little until this wag goes away.

To adjust the rate at which the robot spins in place, adjust the limits, sometimes referred to as ATV or channel travel on the turn channel on your radio. Please note that various mechanical factors will also affect this rate. If your motors are unequal in strength, the gyro will only allow the stronger motor to go as fast as the weaker one.

Wrap-Up

While robot design and construction play a huge role in the outcome of combat robotics matches, a good driver can do very well with a poor robot. While it can't correct for lack of driving practice, it can help to make the robot go where you tell it. If you've got a robot that's a little tricky to handle, give gyros a try. SV

PRODUCT REVIEW — The NPC T-64 Gearmotor

● by Michael "Fuzzy" Mauldin, Team Toad

The simplest way to get your robot moving is with a gearmotor. That's an electric motor combined with a reduction gear in a single package. For a large robot (120-340 pounds), our favorite is the NPC T-64 (formerly known as the 64038).

The output shaft is short and fat with four 5/16-24 bolt holes. NPC sells an aluminum hub that mates this shaft to their line of foam-filled rubber tires. The tires come in 10", 12", and 14" diameters. The gearbox mounts easily with four 5/16-24 bolt holes and two 5/16 through holes in the plate.

There are two major options for this motor: an optional 14-1 gear set increases output RPM by 43%. We prefer the stock 20-1 reduction, with lower gearing, for better acceleration and reduced current draw.

We do use the second option: a heavy-duty gearbox plate made from billet aluminum instead of the stock cast part. On a superheavyweight robot, the impact forces from combat can crack the mounting holes on the stock plate.

Six of these motors powered IceBerg — our superheavyweight BattleBots quarterfinalist — and two powered IceCube — a middleweight that took second place at RoboGames this year.

Although rated at 24 volts DC, we run the motors at 36 volts using two 3.6 Ah Nicad battery packs and one Vantec RDFR36e speed controller for each pair of motors.

With 20-1 reduction and 10" tires, the top speed is 19 mph. We've been able to outpush opponents for three minutes without once having an overheated motor or speed controller failure.

Bottom line: The T-64 is strong, fast, battle-tested, and easy to use. Get yours from NPC at **www.npc robotics.com**. SV

EVENTS

UPCOMING — October and November

House Of Robotic Destruction — This event will take place in Fall of 2006 on October 14 in Cleveland, OH. It is presented by the Ohio Robotics Club.

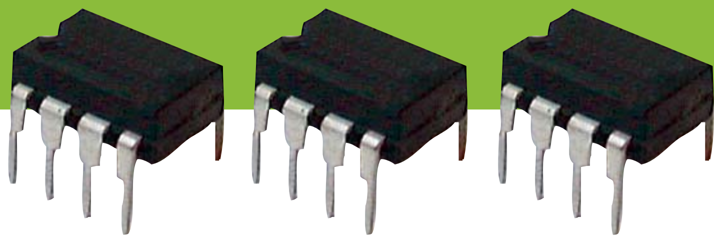
The Ohio Robotics Club will be holding its third House Of Robotic Destruction event at the Olmstead Falls Community Center, just outside of Cleveland; 150 gram Flea, 1 lb Ant, and 3 lb Beetleweight double-elimination tournaments will be run. Cost is $10 per bot. The ORC insect arena is 4' x 8' in size, halfway through a match two 14" x 14" pits open. Further information can be found at **www.ohiorobotclub.com**.

O.R.C.
Ohio Robot Club

Halloween Robot Terror — This event will take place on October 28 in Gilroy, CA. It is presented by California Insect Bots.

Venue is Hobby World — which is located at 6901 Monterey Rd., Gilroy, CA 95020. Bot Gauntlet Baron Dave says "This is open to Fleas, Ants, and Beetles. The only rule I have for the bot costume contest is you must use the bot you brought to fight with. You will NOT be fighting with your bot costume on your bot. Builders and team mates are more than welcome to wear a costume to this event, but please remember to make it safe for anyone that's working on the bots. Weigh-in starts at 10:00 AM and fighting starts around Noon. The entry fee will be $20 per bot with prizes for 1st, 2nd, and 3rd place in each weight class. For fight rules, go to **www.sacbots.com/eventrules.html**. SV

ROBOT's Little Helper



by Ron Hackett

Wouldn't it be great if there were a simple eight-pin microprocessor that could be programmed in Basic to perform various I/O functions and thereby relieve your robot's already over-burdened CPU? Even better, imagine being able to buy such a chip for less than four dollars! That way you could easily add one, two, or more of these "little helpers" to your bot and implement sophisticated sensory functions that would otherwise overwhelm a single, Basic-programmed CPU.

Well, believe it or not, such a chip already exists — it's the PICAXE-08M processor. Produced by Revolution Education — a British company dedicated to promoting robotics in primary and secondary education — the PICAXE-08M is actually an eight-pin PIC chip with a built-in Basic interpreter. The 08M is one of a line of Basic-programmable processors that range in size from eight pins to 40 pins, and include a surprisingly sophisticated range of Basic commands.

For complete documentation on the hardware and software specifications of the PICAXE processors, visit the PICAXE website at **www.picaxe.co.uk** and download all three sections of the PICAXE Manual; they contain a wealth of information to get you started. A Summary of PICAXE Basic commands is also available on my website at **www.RonHackett.net**.

PICAXE Basic is very similar to many other implementations of Basic, including Parallax's BASIC Stamp version, so it is a fairly simple language in which to program. Refer to Section 2 of the PICAXE Manual for a complete description of the Basic commands and programming environment. Many of the 08M's built-in commands are particularly useful in the field of robotics, especially the following:

• infrain2 — Receives and decodes an infrared signal from another PICAXE chip, or an ordinary TV remote control.

• infraout — Transmits an infrared signal to another PICAXE chip (or to a TV, stereo, etc.).

• pwmout — Produces a continuous PWM output in the background for DC motor control (or IR object detection, as we will see later), freeing your program to carry out other tasks simultaneously.

• readadc10 — Performs a 10-bit analog to digital conversion.

• serin – Receives five-volt level serial input at various speeds.

• serout — Transmits five-volt level serial output at various speeds.

• sertxd — Provides serial output to the Programming Editor's "debug" screen, which is very helpful when debugging your code.

• servo — Produces a continuous output in the background to drive a radio-control style servo motor, freeing your program to carry out other tasks simultaneously.

• setint — Enables interrupts on specific input conditions.

A brief summary of the features of selected PICAXE processors is presented in Figure 1. As you can see, the 08M can store a program of approximately 80 lines of Basic code, which is more than enough for just about any robot I/O function. The Integrated Development Environment (IDE) is available free from the PICAXE website. It has a fairly full-featured graphic user interface and runs on Windows 95, 98, ME, NT, 2000, and XP. Unfortunately, a Macintosh version of the software is not available, but the new Intel-based Macs can run Windows XP, so it should be possible for Mac users to join in on the fun anyway!

	PICAXE-08M	PICAXE-18X	PICAXE-28X	PICAXE-40X
Pins	8	18	28	40
I/O Pins	5	14	21	32
Input Pins	1 - 4	5	0 - 12	8 - 20
Output Pins	4 - 1	9	17 - 9	17 - 9
ADC Pins (10-bit)	3	3	4 - 0	7 - 3
PWM Pins	1	1	2	2
Program Memory	80 BASIC lines	600 BASIC lines	600 BASIC lines	600 BASIC lines
Storage variables	48 bytes	96 bytes	112 bytes	112 bytes
Data Memory	256 - Program	256 bytes	128 bytes	128 bytes

FIGURE 1. Features Summary of Selected PICAXE Processors.

Figure 2 shows the pinout of the PICAXE-08M. The first thing you may notice is the somewhat confusing usage of the word "pin." What are commonly called external "pins" are labeled "legs" in the PICAXE documentation. This is because "pin" is a keyword in the PICAXE Basic programming language. In the PICAXE documentation, which is primarily written for school children, Revolution Education did not want to use the same word in two different ways. So the physical external pin is simply referred to as a "leg." Therefore, Figure 2 shows the I/O port pins matched to the external pin (leg) numbers. [The author thanks Clive Seager of Revolution Education for his helpful clarification of this issue.]

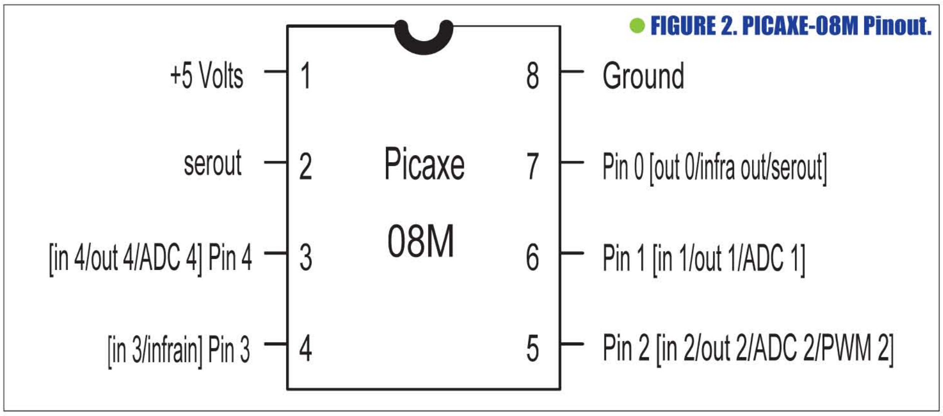


FIGURE 2. PICAXE-08M Pinout.

In any case, the PICAXE-08M has five general-purpose input/output pins. I/O pin 0 (external leg 7) is output only, I/O pin 3 (external leg 4) is input only, and the other three I/O pins (1, 2, and 4 — external legs 6, 5, and 3) can be individually programmed as inputs or outputs. As shown in Figure 2, all of the I/O pins have multiple functions, which are thoroughly discussed in the PICAXE manual.

The second thing worth noting about Figure 2 is the absence of dedicated crystal or resonator pins. The 08M has a built-in 4 MHz resonator which is switchable to 8 MHz under program control. This is a good thing, since it frees up two of the 08M's pins for general-purpose I/O.

On the down side, the internal resonator is not as accurate as the external one found on many of the larger PICAXEs, but it is accurate enough for the types of tasks called for in typical robotics applications. Also, there is a Basic command (calibfreq) that allows you to fine-tune the 08M's operating frequency for increased accuracy. Of course, you will need a frequency counter or oscilloscope to determine when you have it right.

Programming a PICAXE

Figure 3 presents a complete schematic of the three-wire PICAXE programming interface, which only requires two resistors (R1 and R2). An optional resistor on the serout line (R3) provides short-circuit protection. As you may have noticed back in Figure 2, the serial output pin can also function as a general-purpose output or an infrared signal output for your program. If your output is as simple as an LED, you can leave it and the programming cable connected at the same time. All that will happen is the LED will flicker during program download. However, if you are using output 0 for something more involved (e.g., a motor), it's best to disconnect your output during program download.

A simple switch – and a better memory than I have — is all it takes. To avoid the hassle, I try to use output 0 for an LED whenever I can. A special programming cable is available from Revolution Education, but it is easy to wire a standard RS-232 nine-pin connector with the three required lines, as Figure 3 demonstrates.

"Hello World" Circuit and Program

To get a sense of just how easy it is to program a PICAXE-08M, we are going to start by breadboarding the "hello world" circuit presented in Figure 4 and photographed in Figure 5. At this point, you either follow along as an intellectual exercise, or first purchase a couple of PICAXE-08Ms (see the Sources sidebar). Also, download the free PICAXE Programming Editor software from Revolution Education. There isn't space here to go into detail on how to

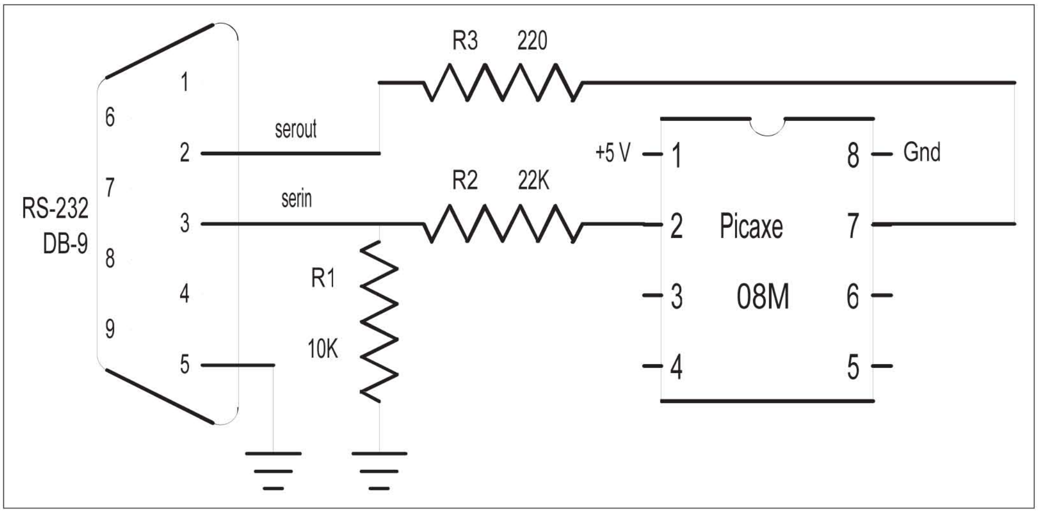


FIGURE 3. PICAXE-08M Programming Circuit.

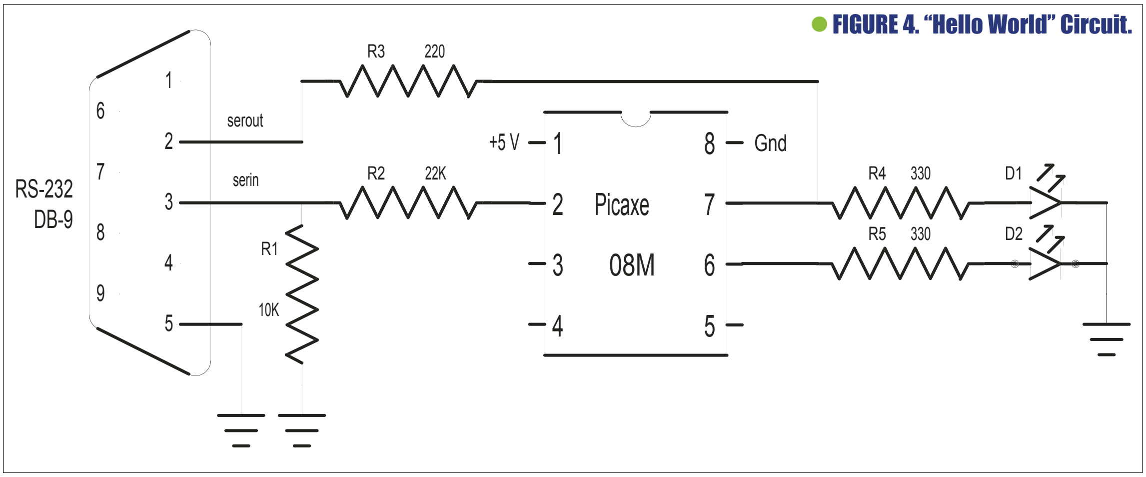


FIGURE 4. "Hello World" Circuit.

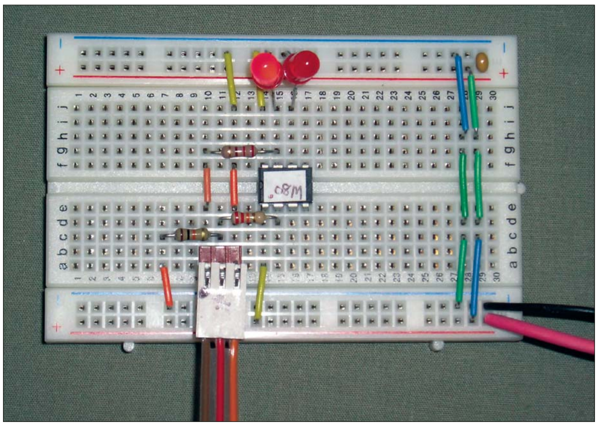


FIGURE 5. "Hello World" Breadboard.

use the software, but it's very intuitive and includes ample documentation.

For our simple "Hello World" example, we are going to program the 08M to alternately blink two LEDs on I/O pins 0 and 1 (external pins 7 and 6). Since all PICAXE chips are actually PIC chips, their output pins are capable of sourcing or sinking a maximum of 25 mA, so LEDs can be directly driven as long as an appropriate current-limiting resistor is included in the output circuit. Actually, if you look closely at the photograph in Figure 5, you won't see the current-limiting resistors because they are built into the LEDs I am using, which makes breadboarding a little simpler. If you are interested in these LEDs, see the Sources sidebar.

The actual "Hello World" Basic program (presented in Figure 6) is very simple and can be quickly typed into the Programming Editor software directly. As you can see, it's very similar to standard Basic. In case you have been wondering, you can power the circuit of Figure 4 using a regulated five volt supply, but it will also function perfectly well with a supply consisting of three AA alkaline batteries.

Just out of curiosity, I have been running my battery-powered "Hello World" setup for more than 500 hours and it's still going strong, so an extra 08M or two certainly won't overly tax your bot's batteries. Of course, if other components in your circuit require a regulated five-volt supply, you don't have the choice.

Infrared Obstacle Detection Circuit and Program

Now that we have a basic understanding of how to set up and program a simple PICAXE-08M circuit, let's try something that will be more helpful to a robot. One of the PICAXE Basic commands mentioned earlier is "pwmout," which produces a continuous PWM output in the background. Therefore, your program can initiate the PWM output and go on to attend to other tasks without the hassle of having to repetitively produce each individual pulse — a tremendous code simplification, to say the least.

Of course, the pwmout command is ideally suited for DC motor control. A PICAXE-08M and an H-bridge chip, such as the L293D, are essentially all you need for full PWM control of one DC motor. If you want to control two DC motors (and what robot builder wouldn't), you will need two 08Ms or one of its bigger siblings (the 28X or the 40X), both of which have two independent PWM outputs and more than enough computing power to also be your bot's CPU.

I have built a couple of robots that use a PICAXE-28X as the CPU and an 08M as an I/O "helper." The two PWM outputs on the 28X provide full proportional control for two DC motors, and the 08M assists by providing left and

```
;===== HelloWorld.bas ===========================================

;       This program runs on a PICAXE-08M processor at 4 MHz.
;       It alternately blinks LEDs on outputs 0 & 1 (pins 7 & 6).

;===== Constant Definitions =====================================

        symbol LED0 = 0          ; LED on output 0 (pin 7)
        symbol LED1 = 1          ; LED on output 1 (pin 6)

;===== Main Program =============================================

main:   high LED0                ; LED0 on
        low  LED1                ; LED1 off
        wait 1                   ; wait 1 second

        low  LED0                ; LED0 off
        high LED1                ; LED1 on
        wait 1                   ; wait 1 second

        goto main                ; do it again, forever
```

FIGURE 6. "Hello World" Basic Program.

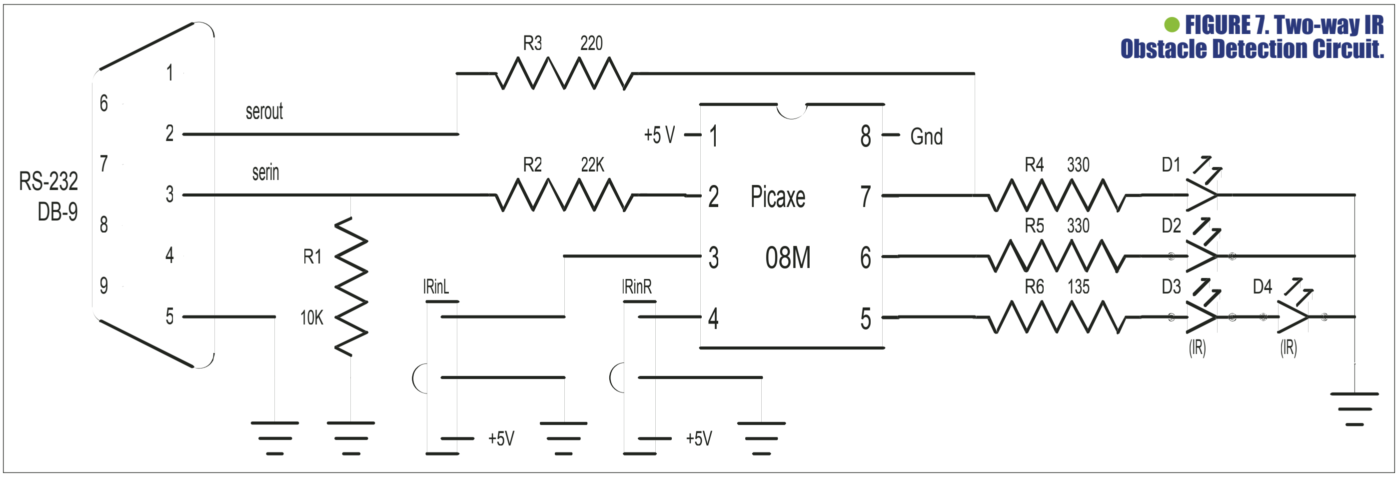


FIGURE 7. Two-way IR Obstacle Detection Circuit.

right non-contact IR obstacle detection.

As a demonstration of the I/O capabilities of the PICAXE-08M, let's take a look at one possible IR obstacle detection circuit, presented in Figure 7 and photographed in Figure 8. As you can see in the photograph, heat-shrink tubing protects the IR LEDs from extraneous light. There are several simple IR receiver modules on the market that operate at 38 kHz, and the 08M's on-board PWM circuitry can easily be used to generate the 38 kHz pulses necessary to drive an IR LED (see the Sources sidebar).

As mentioned earlier, the 08M's output pins are capable of directly driving an IR LED. In fact, one output pin can drive two IR LEDs connected in series as shown in Figure 7. The two IR LEDs I used (again, see the Sources sidebar) each drop 1.2 volts, which leaves 2.6 volts across the 135 Ω resistor (actually two 270 Ω resistors in parallel). As a result, the current is approximately 19 mA which is well within the output pin's specifications. You may want to breadboard just this part of the circuit first, and determine the current for your specific components before connecting the IR LEDs to the 08M's output pin, just to be safe.

The 08M's two outputs could be directly connected to two input pins on your main processor. However, as a safety precaution, you should include a 1K resistor in each connection, just in case one of the CPU's inputs accidentally gets reprogrammed as an output, which could result in a direct short in the line and a possible dead I/O pin. The LEDs can even be left in the circuit if you would like visual feedback on the circuit's functioning. In this way, your bot's CPU can be relieved of the processing overhead required to carry out this function, and it only takes two of its I/O pins to do so.

Figure 9 presents the Basic program for the obstacle detection circuit, which takes up less than 15% of the program space in the 08M. Essentially, the program consists of a simple loop that outputs a burst of PWM pulses, turns off the pulses, checks for left and right echoes, and displays the results on the two LEDs. As you know, interpreted Basic is a relatively slow language, so the program uses two different techniques in order to be fast enough to detect the echo.

First, the setfreq command at the beginning of the program shifts the 08M's internal oscillator from 4 MHz to 8 MHz, because Basic just isn't fast enough to catch the echo at 4 MHz. Some care must be exercised when using the setfreq command. It's easy to lose communications with the programming software unless you also change its setting to reflect the new oscillator speed; be sure to read the PICAXE manual on this issue.

Secondly, the PICAXE Special Function Variable "pins" allow direct access to the I/O port register of the chip, which is a very powerful and useful feature of the PICAXE system — you can refer to the documentation for a complete explanation.

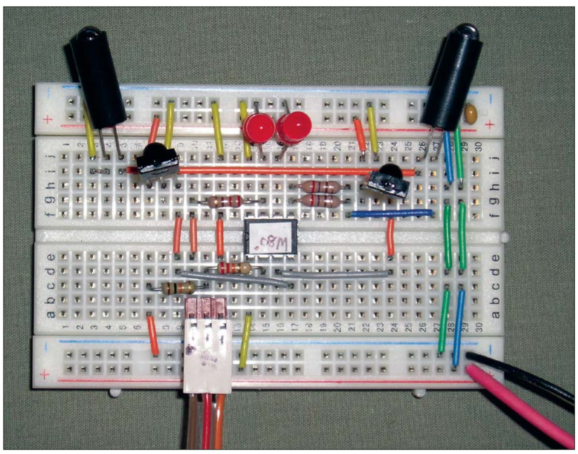


FIGURE 8. Two-way IR Obstacle Detection Breadboard.

Through experimentation, I found this approach to be much faster than the alternative of testing the IRinL and IRinR symbols (defined at the beginning of the program) with if-then or branch commands. Notice that four of the five constant definitions are not used anywhere in the program. They could have been removed entirely, but you may want to do a little experimenting yourself, so I left them in the program.

The Panasonic IR detectors in the circuit are active low devices. The comments in the program explain how

SOURCES

The following items are available from the author:
www.RonHackett.net

- PICAXE-08M
- Five-volt resistorized LED
- IR LED
- Panasonic PNA4602M IR Detector

PICAXE-08Ms can also be purchased directly from Revolution Education:
www.picaxe.co.uk

```
;===== IR_detect.bas ===============================================================

;       This program runs on a PICAXE-08M processor at 8 MHz.
;       It emits 38 kHz IR PWM bursts & checks for echoes after each burst.

;===== Constant Definitions ========================================================

        symbol  LED_L = 0               ; Left visible LED on output 0 (pin 7)
        symbol  LED_R = 1               ; Right visible LED on output 1 (pin 6)
        symbol  IRout = 2               ; Both IR LEDs on PWM output 2 (pin 5)
        symbol  IRinL = input3          ; Left IR detector on input 3 (pin 4)
        symbol  IRinR = input4          ; Right IR detector on input 4 (pin 3)

;===== Variable Definitions ========================================================

        symbol  echo  = b0              ; used to store & manipulate echo values

;===== Main Program ================================================================

        setfreq m8                      ; set internal resonator to 8 MHz

        dirs = %00000111                ; configure I/O pins 2, 1 & 0 as outputs

main:   pause 200                       ; pause for 100 mS
                                        ; (duration is halved @ 8 MHz)
        pwmout IRout, 50, 106           ; PWM burst @ 38 kHz, 50% Duty Cycle
        pause 20                        ; burst duration = 10 mS
                                        ; (duration is halved @ 8 MHz)
        pwmout IRout,  0, 106           ; PWM off

        echo = pins                     ; move IR inputs to bits 4 & 3 of echo
        echo = echo / 8                 ; and shift them right to bits 1 & 0

                                        ; IR detectors are active-low inputs,
        pins = NOT echo                 ; so invert echo for active high
                                        ; and move it to outputs 1 & 0

        goto main                       ; do it again, forever
```

FIGURE 9. Two-Way IR Obstacle Detection Basic Program.

the inputs are converted to active high for output. The detectors also specify 4.75 volts as the minimum operating voltage. In spite of that, the circuit operates reliably with the 4.5 volt (nominal) battery pack, and provides a range of about 12 inches. I'm sure this would deteriorate rapidly as the batteries discharge. Of course, a regulated five-volt supply provides improved functioning of the circuit.

Conclusion

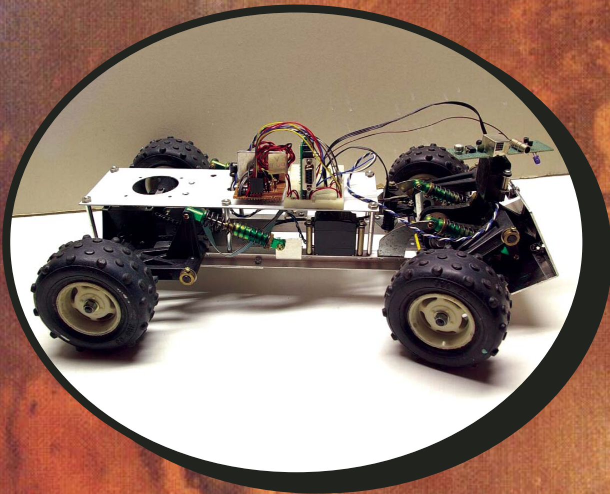
I hope this article has peaked your interest in the PICAXE-08M as a "little helper" for your robots. We have barely scratched the surface of the capabilities of this amazing little chip. I didn't even mention that the "M" in PICAXE-08M stands for "music" — the 08M has built-in commands for producing complete tunes with surprisingly little programming. So, if you would like your bot to "whistle" while he or she works, the 08M is just the chip for you!

The PICAXE-08M can be very helpful as an interface chip between your robot's CPU and any sensor or output that requires anything but the simplest of interfaces. To give just one of many possible examples, consider the SRF005 Ultrasonic Range Finder. It is very popular with roboticists, but it requires a fair amount of attention from your bot's CPU (i.e., frequent trigger pulses followed by relatively long periods of "listening" for an echo). An 08M could easily handle this task and inform your bot of impending danger, while greatly simplifying your CPU's main program.

The PICAXE-08M packs a tremendous amount of computing power in a very small package and at a surprisingly affordable price. If you are willing to give up PWM motor control and settle for the "full-speed ahead" variety, I'm sure you could build a fully-functional autonomous bot using one PICAXE-08M as its only processing power. In fact, I have already started working on just such a bot! SV

ABOUT THE AUTHOR

*You can reach Ron via email at RonHackett@verizon.net or visit his website at **www.RonHackett.net**.*



Do-It-Yourself MARS ROVER

● by Dan Gravatt

Give yourself a "mission" to build your own Rover with parts you already have at home.

I saw a sign a couple of years ago that said "Electronics Garage Sale" and nearly caused an accident slamming on the brakes in my truck. It was worth the stop, because they were selling a late 1970's vintage radio controlled dunebuggy powered by a small glow-fuel engine for $20. The engine was shot but the electronics were okay, and my plan was to remove the servos and receiver and junk the rest of it.

Once I got home, I scavenged the electronics and discarded the engine, fuel tank, and muffler. All that was left was the chassis, suspension, and drivetrain — all beat up and covered with glow fuel residue. It was a mess, but it was too good to just throw away, so I set it aside and moved on to other projects. Several projects later, after accumulating some nice stepper motors and other bits, I decided it was worth the (substantial) effort to clean up the parts and use them as a robotics platform. Although it's very unlikely you will be able to exactly duplicate my "Mars Rover," the electronics and code described here should be compatible with pretty much any four-wheeled chassis you would care to use.

Design by Parts

The designs of my projects are generally dictated by the parts I have, and this one is no exception. I have a hard time buying parts when I already have lots and lots of them around. So the only reason I used a bipolar stepper drive motor was that I had a good one with ball bearings and a shaft diameter which fit the drive sprocket from the dunebuggy. The BASIC Stamp 2 controller I used is a bit underpowered for this application, but it was available. I'm pretty happy with the results even though I had only a vague idea what it would look like when I started building.

The mechanical design of the Rover is pretty straightforward and not all that different from the original dunebuggy. The suspension and steering mechanisms are pretty much unchanged. I had to replace all of the original frame with aluminum C channel stock, and I converted the chassis from four-wheel drive to rear-wheel drive by shortening the drive chain. I mounted the stepper drive motor and steering servo on the new frame, and the result is shown in Figure 1.

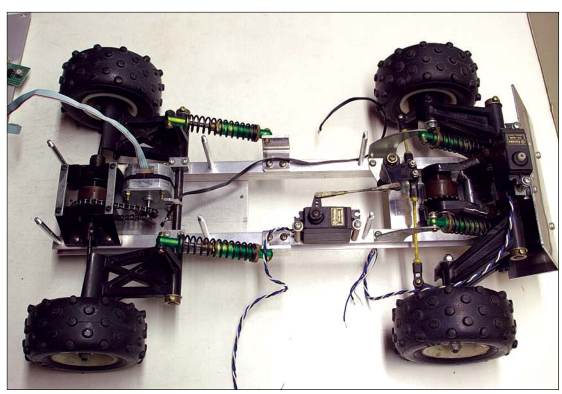


Figure 1. Mechanical components of the Rover.

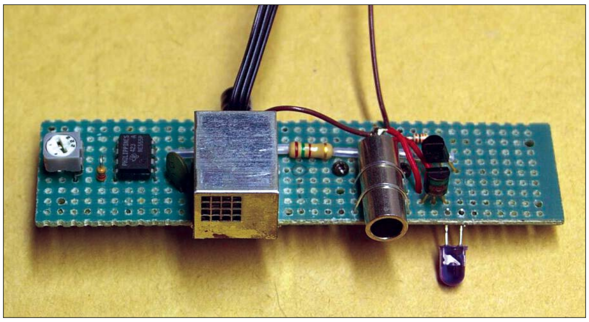


Figure 2. Closeup of the IR sensor board.

Sensor Selection

Once the mechanical platform was finished, I started thinking about what sensors to use so that the Rover could navigate itself autonomously. I didn't have one of those nifty ultrasonic range finder sensors, so I settled for a low-tech infrared obstacle detector using a 38 kHz infrared receiver module from an old VCR. The simplicity of this sensor is compensated by the way the BS2 code divides the "landscape" in front of the Rover into five horizontal sectors, as well as one "look-down" sector to prevent it from driving off a cliff.

The IR sensor board (see Figure 2) is mounted on a servomotor on the front of the Rover for a good field-of-view, and pans back and forth over an arc of 135 degrees or so while the Rover is in motion. The infrared LED used for the horizontal sector scanning is mounted in a metal tube to better direct the scanning beam and minimize any stray light hitting the detector.

A second LED for the "look-down" function is mounted separately and angled down and away from the detector. A control input from the BS2 (see Figure 3) selects one of the two LEDs at different times during each sweep. I found that the 100 µF capacitor on the output of my IR sensor was necessary to give a stable, glitch-free output to the BS2.

Since even the best obstacle sensors can miss things, I wanted a back-up ability to sense if the drive motor stalls due to hitting something. I glued a bar magnet to the end of the drive sprocket and positioned a Hall-effect sensor (scavenged from an old 3.5" floppy drive) near it to monitor the North-South field transitions as the sprocket turns (see Figure 4). The sensor is simple, but the code processing its output is not (more on this later).

Navigation

The Rover knows what's ahead of it; now it needs to navigate itself. A servomotor steers the front wheels, and the rear wheels are driven by the bipolar stepper motor. Along with panning the sensor servomotor and collecting data from the IR sensors and the Hall-effect sensor, this is too many tasks for the humble BS2 to handle. Controlling the bipolar stepper motor is the most time-consuming of these tasks, so I added an EDE1204 bipolar stepper controller to the design to lighten the BS2's workload. The EDE1204 is a Microchip PIC16C54B which the folks at E-Lab have programmed to control a bipolar stepper motor with a fair amount of flexibility and a minimum of outside help.

The control outputs from the EDE1204 are fed to an L298N dual H-bridge which drives the stepper motor coils (see Figure 5). Schottky diodes on the motor outputs protect the L298N from voltage spikes when the coils are de-energized. The two one-ohm resistors sense and limit the current supplied by the L298N, but I was unable to find a formula on the datasheet to calculate exactly what the current limit is for a given resistor value.

On the other hand, I don't know the specifications on the bipolar stepper motor I'm using either, as it was scavenged from an inkjet printer. It moves the Rover around just fine with a 12-volt supply, such as a small lead-acid gel cell.

Several of the control inputs to the EDE1204 are wired high or low to set its basic operating parameters. I have configured it to generate the motor control pulses by itself in full-step (rather than half-step) mode. It is also set to keep the motor coils energized even when stopped, so the Rover won't roll down a slope. The remaining functions required for the Rover's operation (motor start/stop, direction, and speed) are under the BS2's control.

The BS2, of course, is where everything comes together (see Figure 6).

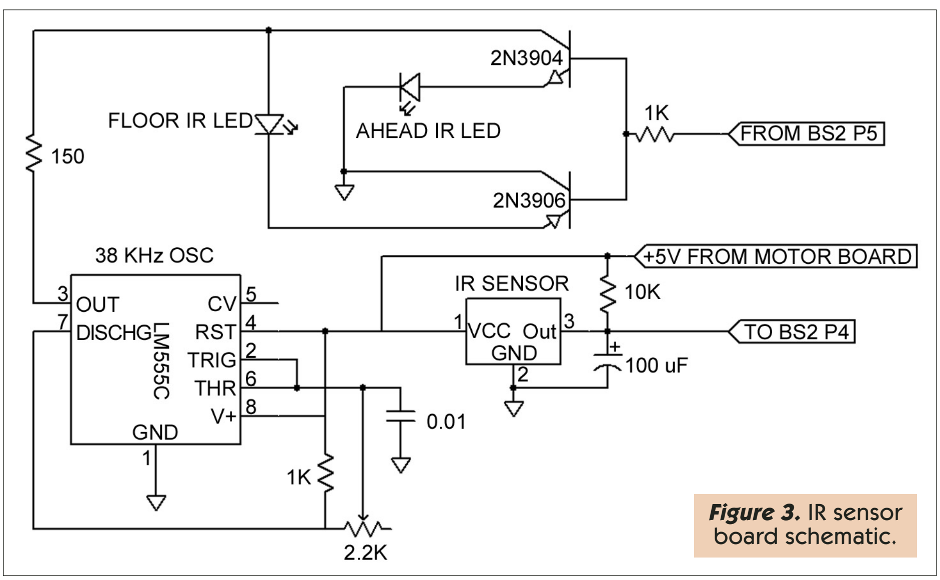
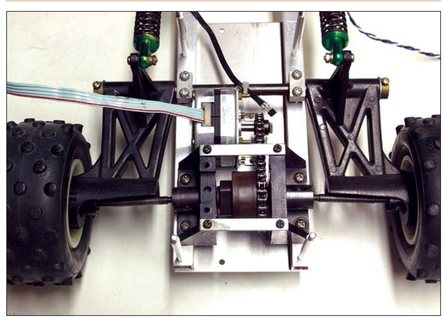


Figure 3. IR sensor board schematic.

Figure 4. A closeup view of stall sensor and magnet.



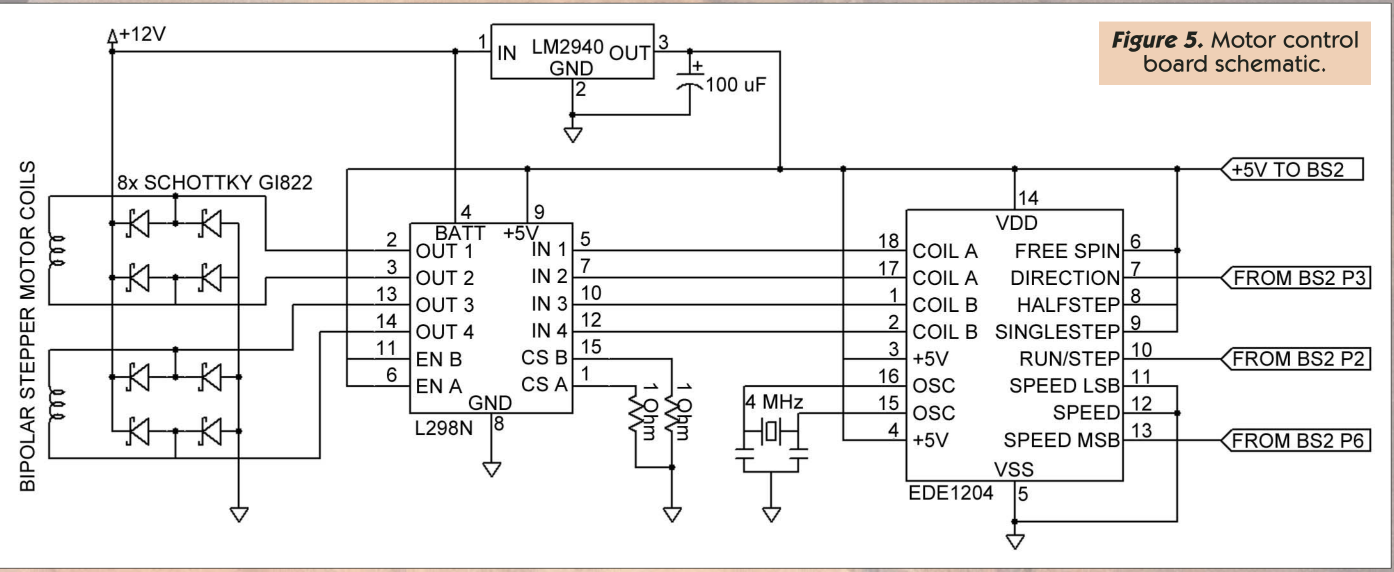


Figure 5. Motor control board schematic.

Nothing too fancy in this part of the circuit, and only half of its I/O pins are used for this application. Less than half of its two kilobytes of memory is used for the code.

Electronics Construction

Construction of the electronics for the Rover is pretty straightforward. I suggest building and testing the stepper motor control board first, because a Rover that can't move isn't much good. Check the wiring to your stepper motor to ensure that the direction input for the EDE1204 corresponds to the direction the Rover actually moves. Then, build the IR sensor board and adjust the oscillator frequency with the variable resistor until it matches the IR sensor you use. Check that the IR sensor provides a stable output when an obstacle is present or absent, and don't forget to check the "look-down" LED too.

The BS2 code expects a logic low output from the sensor when an obstacle is present, so if your IR sensor's output is inverted, you will need to modify the code or re-invert the sensor output. Finally, connect everything to the BS2 and load the code.

Load the Code

And how about that code, anyway? That's where the behavior of the Rover is created, and the code described below (which is available on the *SERVO* website at **www.servomagazine.com**) is only one way to use the hardware I've described above. If you've got more processing power onboard and/or better programming skills than I, very complex behavior can be created using just a few sensors. Note that my code uses the PBASIC 2.5 version of Parallax's tokenizer to implement true IF-THEN-ELSE branching.

Most of the time the BS2 spends in the Rover code is in the subroutines called MOVING and SWEEPBACK. After a brief power-on pause to connect the battery to the Rover, the MOVING subroutine is the first code to run. It does several interleaved tasks:

- Checks the direction of the last sensor servo sweep (the first sweep is always left to right).

- Pans the sensor servo from full left to full right.

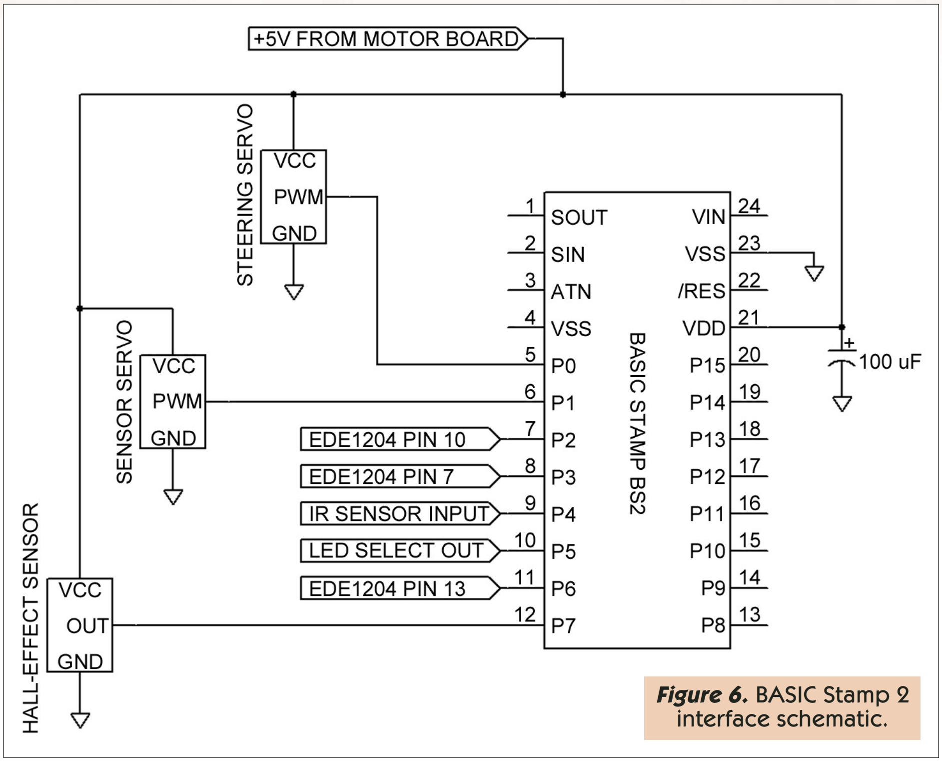


Figure 6. BASIC Stamp 2 interface schematic.

About the Author

Dan Gravatt is a licensed geologist in the state of Kansas. He can be reached at dgravatt@juno.com

• Collects data from the IR sensor at five points (full left, left-center, center, right-center, full right) during the sweep.

• Checks for the presence of a drop-off (such as a table edge) directly in front of the Rover.

• Detects magnetic field transitions from the stall sensor and increments a counter (STALLCOUNT).

• Refreshes the steering servo to maintain its position.

• At the end of the subroutine, increments a counter bit (SERVTURN) so that the next sensor servo sweep will be from right to left.

The SWEEPBACK subroutine does the same tasks as the MOVING subroutine while sweeping the sensor servo from full right to full left.

The MAIN subroutine is where the Rover's behavior is determined. Here, the data collected from the latest IR sensor sweep and the stall sensor is used to select the appropriate direction of travel. Data which potentially represent the greatest hazards to the Rover (drop-off or obstacle directly ahead, or motor stall) are evaluated first, and the Rover will back up if any of these conditions exist.

Motor stall is detected by counting the magnetic North and South (0 and 1) states from the stall sensor during an IR sensor sweep, storing that count, and comparing it to the count from the subsequent IR sensor sweep. If the two totals are the same, stall is detected and the Rover backs up to clear the obstacle. These totals are almost never the same during normal driving because the rotation of the drive sprocket is not synchronized with the stall sensor count interval, so the probability of a false stall signal is low. On the downside, it takes up to two full sensor sweep periods for the Rover to detect a stall.

In the event that the Rover gets itself stuck in a place where it cannot move forward due to an object directly ahead (detected by the IR sensor), and cannot back up due to an object behind (detected by the stall sensor), the Rover will stop moving. This condition is the first one checked in the MAIN subroutine because it combines the two hazard conditions described above. Evaluating either hazard separately before evaluating them together might overlook the Rover's true situation and cause it to continue backing up into an obstacle. The IMSTUCK subroutine stops the Rover and could also include some code to provide an indication that the Rover needs help.

To avoid what Douglas Adams (author of the *Hitchhiker's Guide to the Galaxy* books) would call a Herring Sandwich Loop, the Rover does not back up in a straight line. Instead, the BACKUP subroutine will turn the steering servo all the way to one side and turn while backing up. If more than one interval of backing up is required to clear an obstacle, the BACKUP subroutine will alternate the direction of the turns using a counter bit (BACKTURN). This will allow the IR sensor to sweep the terrain to the left and right of an obstacle and (hopefully) find a clear path forward.

If the MAIN subroutine determines that there are no immediate hazards to the Rover, it next evaluates the IR sensor data for obstacles to the left or right of the direction of travel. Obstacles close to the direction of travel result in a sharp turn away from the obstacle, while those detected farther from the direction of travel results in a gentle turn away from the obstacle.

By not overreacting to obstacles to the sides, the Rover can maneuver better in tight quarters and maintain a smoother direction of travel while making it less likely that an obstacle will eventually be met "head-on." If the Rover detects obstacles to the far left and far right (such as in a hallway), but nothing ahead, or if no obstacles are detected by the IR sensor, the Rover will continue in a straight line.

The remaining subroutines are executed between sensor sweeps to configure the steering servo and the EDE1204 to move the Rover in the direction selected by the MAIN subroutine during the next sensor sweep. The steering servo is set first, then the appropriate speed and direction of rotation for the stepper drive motor, and the code jumps back to the MOVING subroutine.

Mission To ... ?

Once you have your Rover operational, give it a mission. Place a camera on it for reconnaissance, or a robotic arm to collect items the Rover encounters. Add additional navigation sensors, like an electronic compass, an ultrasonic rangefinder, or a second IR sensor to search for obstacles behind the Rover. Give it the ability to collect data on temperature, radiation, light intensity, or other characteristics of its environment.

Happy Roving! SV

Parts List

- Steerable four-wheeled chassis
- Bipolar stepper motor
- Two standard servomotors
- Small 12-volt lead-acid gel cell
- BASIC Stamp 2
- EDE1204 bipolar stepper motor controller (or use an EDE1200 for unipolar stepper motors)
- 4 MHz ceramic resonator with capacitors
- L298N dual H-bridge driver
- LM2940CT five volt regulator
- LM555 timer
- 38 kHz IR receiver module
- (2) infrared LEDs
- Hall-effect sensor and bar magnet
- (8) Schottky diodes, GI822 or equivalent
- 2N3904 transistor
- 2N3906 transistor
- (3) 100 µF capacitors
- 0.01 µF capacitor
- (2) 1 ohm resistors
- (2) 1K ohm resistors
- 10K ohm resistor
- 150 ohm resistor
- 2.2K ohm potentiometer

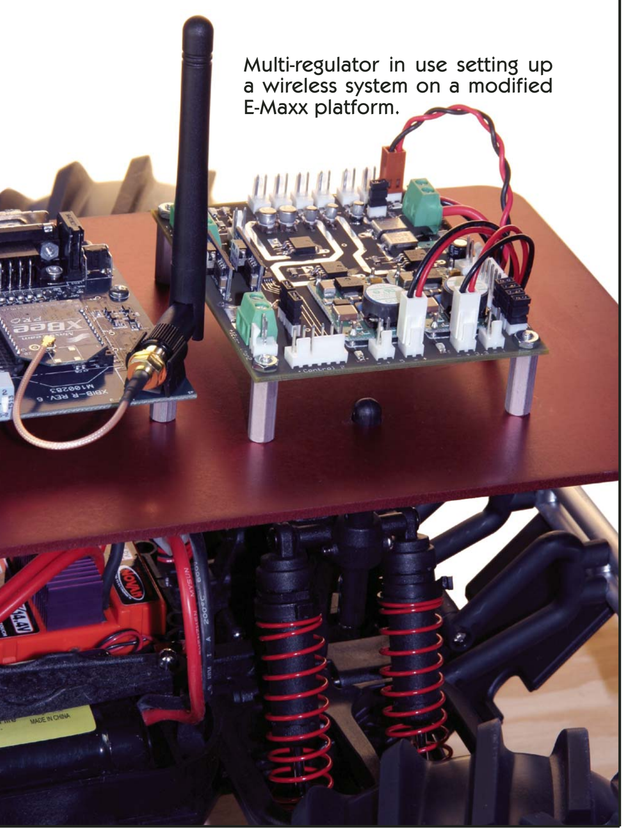
ROBO-LINKS

For the finest in robots, parts, and services, go to www.servomagazine.com and click on Robo-Links to hotlink to these great companies.

Energy Management *for* AUTONOMOUS ROBOTS

by Bryan Bergeron

In robotics — as in biological systems — energy is a finite, precious resource. Even with a lightweight platform, an efficient drive mechanism, and low-power components, most robots exhaust their electrochemical batteries in minutes to hours. While there are exceptional implementations that rely on solar power or fuel cells for extended operation, most robots are truly autonomous for the relatively brief period of time between battery charges.



Multi-regulator in use setting up a wireless system on a modified E-Maxx platform.

Fortunately, there are new battery chemistries on the horizon that use nanotechnology to achieve three times the energy density of conventional lithium ion batteries [1]. Regardless of the chemistry, the energy is finite, and proper energy management techniques and technology can extend autonomous time by a factor of two or more.

This article reviews energy management principles for autonomous robotics, with an emphasis on selecting and designing power supply electronics, how to implement real-time power reconfiguration, and monitoring techniques.

Energy Management

Continuing with the biological analogy, energy management encompasses not only acquiring energy, but regulating its expenditure. A robot that's continually moving and using its myriad sensors to explore the environment will run down sooner than a robot that alters its behavior to reflect the current energy stores. Sleeping when necessary, limiting maximum speed, switching off unnecessary sensors and systems when appropriate, and, for a walking robot, switching to a more efficient gate, can result in significant energy savings.

Power Systems

Power systems — batteries and onboard regulators — are often an afterthought, a distant second or third behind the drive system and the microcontroller — and often for good reason. A typical carpet roamer will do nicely with a few AA NiCds connected to an inexpensive linear regulator, such as the ubiquitous 7805, a few capacitors, and perhaps a zener diode for insurance. This is an appropriate configuration for a simple carpet rover based on a microcontroller and a few proximity sensors.

However, when you start working with one of the new 3.3V multicore processors, one or two 9V wireless pinhole cameras, a 5V 104-format PC card, and a 5V servo controller, all on single 14.4V battery, then power suddenly becomes very important. High current demands, voltage spikes, the need for multiple voltages — some switched on or off during robot operation — are usually beyond the capabilities of a 60 cent linear regulator. Furthermore, when you're working with a $1,200 embedded PC board, $2,800 laser range finder, and another $1,000 in wireless hardware and sensors, the last thing you want is to have your components fry because the circuitry wasn't protected against shorts when the robot flipped.

The optimal mix of battery chemistry and regulator electronics depends on the robot design and application. Given this caveat, my ideal robot power system would have the following characteristics:

- Fused for short circuit protection
- Input polarity reversal protection
- Output voltage surge protection
- Automatic shutdown of motion control functions with loss of remote control signal
- Provide input and output isolation
- Highly efficient
- Affordable
- Lightweight
- Self-monitoring
- Stabile and accurate output
- Support switched power subsystems

REFERENCE

[1] A123 Systems. **www.a123systems.com**.

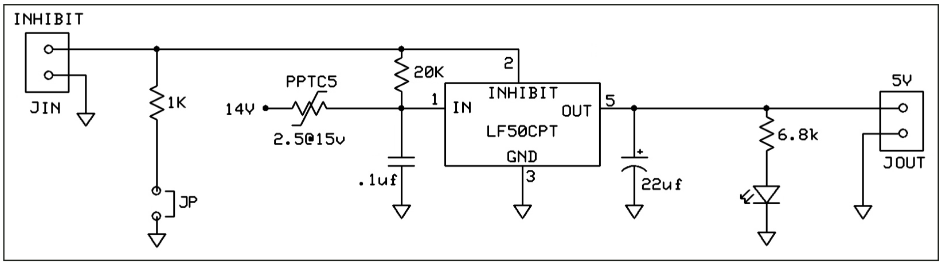


FIGURE 1. Linear regulator with support for subsystem cutoff through the use of the INHIBIT lead.

- Linear ramp-up for servo systems
- LED indicators of operation
- Simple
- Low standby or quiescent power dissipation

Some of these characteristics are mutually exclusive. Even so, let's review some popular regulator technologies, keeping the characteristics of an ideal system in mind.

Linear Regulators

Linear regulators, which act like variable resistors to drop a voltage down to a lower fixed or variable value, are simple, inexpensive, and lightweight. The linear regulator circuit shown in Figure 1 is appropriate for applications with current requirements of 5V at 500 mA or less. The LF50CPT has a very low voltage drop and, more importantly, it supports switched power subsystems through the INHIBIT lead. Driving the INHIBIT lead high by connecting an appropriate source through Jack JIN, turns off any subsystems connected to JOUT. The jumper, JP, is used to keep the regulator switched on. I've used this configuration with good results on a 12-cell NiMH battery pack.

The major downside of this and other linear regulators is efficiency. However, when used to occasionally switch on circuitry, efficiency may not be a major factor.

Switching Regulators

Switching regulators — also called DC-DC converters — are inherently more efficient than linear regulators, but are also more complex, require more components, and are often more expensive. They also tend to be more expensive than simple linear regulators. However, ready-to-use modules, such as those shown in Figure 2, are the quickest route to stable, high-efficiency power.

Switching regulators are more finicky than their linear counterparts, especially when it comes to input voltage. For example, the ETA OC1-3.3 shown in Figure 2 provides 3.3V at 1.5A. While the unit is only about 30 x 20 mm, it's dwarfed by the Bourns SX5A-12 that is capable of supplying any voltage in the range of 0.75 to 5.5V at a full 5A.

Why the size discrepancy? The major difference between the two switching regulators is the range of input voltages they support. The OC1 can work with any voltage from about 10V to 32V — which is perfect for the 12-cell NiMH packs that run 15V to 16V when fully charged. The SX5A, in comparison, expects 10V to 14V. The module shuts down with anything outside of that range.

The efficiency of the OC1-3.3 and 5V OC1-05 is in the low 90s, which is significantly better than anything a linear regulator could provide. In addition, the OC1 series of switching regulators support a remote cutoff for subsystem shutdown. For continuous operation, the remote cutoff (RC) lead is kept high with a pull-up resistor, as in Figure 3.

The basic configuration of the SX5A is similar to that of the OC1. However, the output voltage of the SX5A can be set by either applying a voltage to the TRIM lead of the device or installing a resistor to ground. The value of the resistor is defined by the following equation:

$$R_{trim} = [10.5/(V_{out}-0.7525) - 1] \text{ K ohms}$$

The SX5A, which can be disabled remotely, lends itself to programming, either one time or during robot operation, when combined with a programmable potentiometer, such as the CAT5113. For example, I use the circuit in Figure 4 to supply voltage to a robot arm. Slowly ramping up the voltage at startup avoids the sudden jerking that often occurs when servos are first energized. The CAT5113 is easily controlled

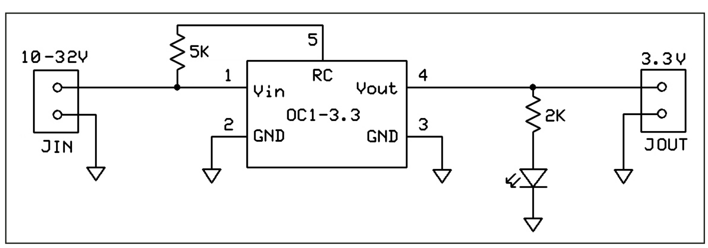
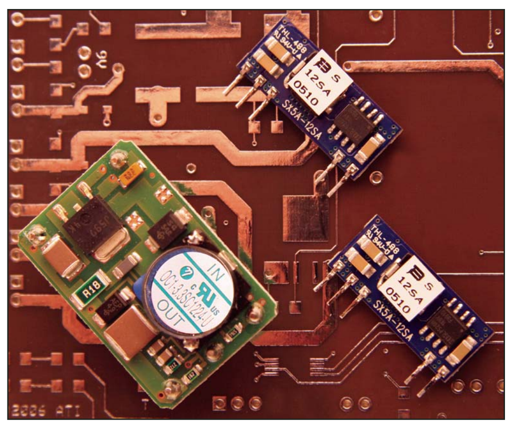


FIGURE 3. The ETA OC1-3.3 configured for continuous operation.

FIGURE 2. Switching regulators. The ETA OC1-3.3 (lower left) provides 3.3V@1.5A, while the smaller Bourns SX5A-12s (right) each provide 0.75-5.5V@5A.



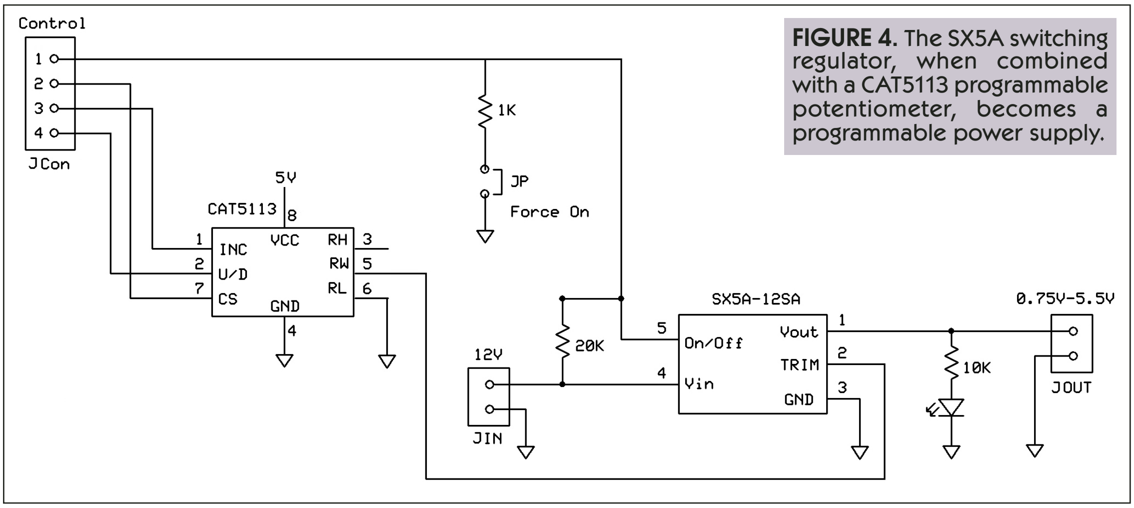


FIGURE 4. The SX5A switching regulator, when combined with a CAT5113 programmable potentiometer, becomes a programmable power supply.

by a Stamp or other microcontroller, and the output pins — High, Low, and Wiper — correspond to the same terminals on a physical potentiometer.

Switched Power

Power to subsystems can be switched, regardless of whether the regulator supports remote operation. Figure 5 illustrates the use of a solid-state relay to switch a 9V supply. I use the circuit to switch between two 9V pinhole cameras on a crawler — one facing front and the other attached to the head of an arm. Using the switch enables me to use one receiver and switch between the two cameras, without the need to handle audio or video signals. The LC110P is capable of switching a continuous current of 120 mA, and 350 mA peak.

FIGURE 5. Solid-state relay used to switch a 9V supply.

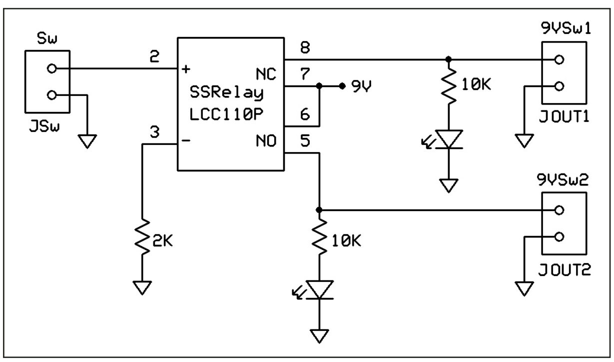


FIGURE 6. Diminutive Polyswitch Resettable Fuse (left) and the ATC Blade Fuse (right) are inexpensive insurance against a catastrophic meltdown.

Safety

Safety should be a major concern when experimenting with a new robot design or even when working on an old design in a new environment. Lithium cells ignite and burn when shorted for only a few seconds, and a $1,200 processor board can be ruined if a stray wire lands on the wrong pad following a crash. Safety involves good design practices, as well as providing technological solutions to inevitable accidents.

Current Limiting

A fundamental safety precaution is to limit the current that can flow from the batteries to the circuitry. A fast-acting fuse — such as the ATC Blade Fuse or a Polyswitch Resettable Fuse — can protect batteries and circuits from disaster.

The advantages of a resettable fuse, such as the Raychem Polyswitch shown in Figure 6 (and previously used in the schematic shown in Figure 1), are size and weight savings, and freedom from having to replace fuses. In addition, instead of designing a board so that the fuse can be easily accessed and changed, a thin Polyswitch can be tucked away out of sight. The price for this convenience is the added resistance of the switch. Whereas an ATC Blade Fuse might have a resistance of a few thousandth's of an ohm, the Polyswitch shown in Figure 6 has a normal resistance of between 0.035 and 0.085 ohms.

Polyswitches are rated by maximum voltage, maximum current, hold current, trip current, and trip time. The Polyswitch in the figure (Mouser 650-SMD250) is rated at 15V, 40A maximum current. It will handle 2.5A indefinitely (hold current) and trip at 5A. The trip time is 25 seconds at 8A.

Current Monitoring

A basic principle in selecting a fuse or resettable fuse is to anticipate the highest normal current and to trip the fuse at anything higher. For example, I use several of the H-bridges from Parallax for motor controllers. The bridges ship with a 25A ATC Blade fuse, but I replaced the fuses with 15A versions after determining that the maximum stall current on my robot was just over 12A. A clamp-on meter, such as the Fluke-337 or Extech 355, is perfect for monitoring high currents without breaking circuit continuity or introducing lead resistance into the circuit.

It's often useful to determine current drawn from a power supply during normal operation, especially during the early design stages. A simple way to monitor current draw in real time is to use a current monitor, which takes the voltage developed across a current shunt resistor and translates it into a proportional output current. This current is then

translated into an output voltage across a precision resistor and ground, as shown in Figure 7. Current through the 0.01 resistor creates a voltage that is sensed by the monitor and a voltage appears across the 10K resistor and ground. Assuming a 0.01 ohm shunt, the relevant equations for the circuit are:

$$V_{sense} = V_{in} - V_{load}$$
$$Vout = 0.01 \times V_{sense} \times R_{out}$$

That is, V_{sense} is the voltage drop across the 0.01 ohm resistor. At 1A, the voltage drop is 0.01V, and Vout is 0.01 x 0.01 x 10K or 1V. A current of 5A will result in 5V across the 10K resistor, which can be fed to the analog input port of a microcontroller for logging or sounding an alarm.

Temperature Monitoring

Overheating of battery packs and regulator boards is a common problem in robotics work. Although most of the switching and linear regulators have some form of thermal auto-shutoff, it helps to know that the temperature is rising well before power is shut off. The least expensive approach to temperature monitoring is to install a thermistor directly on the regulator board, and to periodically poll the resistance with a microcontroller.

The primary advantage of using a thermistor — a temperature sensitive resistor — is sensitivity. Thermistors are more sensitive than thermocouples and most other temperature measuring components. Moreover, thermistors are small, lightweight, and inexpensive. They're also highly non-linear. Fortunately, a simple work-around is to use an ordinary resistor in parallel with the thermistor. Sensitivity suffers somewhat, but linearity is improved.

Most manufacturers provide tables of resistor values for linearization, but a simple heuristic is to use approximately equal values at the desired temperature. For example, the thermistor in Figure 8 has a resistance of 10K at room temperature. When paired with an ordinary 10K resistor, the pair will provide a relatively linear response ± several tens of degrees around room temperature. It's important to note that for this application, the absolute temperature isn't

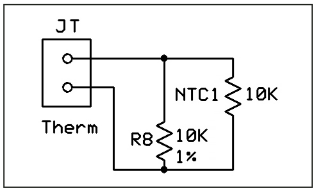


FIGURE 8. Temperature monitoring with a thermistor in parallel with a resistor.

important. What matters is that the temperate has risen above a predetermined level, such as 60° C. You can use a heat gun and thermal probe to determine the resistance of the thermistor/resistor combination at the temperature limit and program your microcontroller accordingly.

A commercial solution to real-time monitoring of temperature, current, and voltage is available from Eagle Tree Systems, in the form of their Micropower E-Logger. The 17 gram system sells for about $100 when configured with sensors and cables for RPM, voltage, current, and temperature. The most impressive part of the package is a Windows-based application that can be used to graphically display the data.

Connectors and Cables

An often overlooked aspect of energy management is the cables and connectors used to carry current from the battery to circuit boards and between circuits. As a rule of thumb, use 14 gauge wire up to about 30A, 10 gauge up to 45A, and eight gauge up to 75A. Of course, eight gauge copper wire is heavier and more difficult to handle than 14 gauge wire, but at 50A or 60A, the electrical difference is significant.

Some of the best connectors available for robotics applications are the W.S. Deans Ultra-Plug, available from Tower Hobbies, and the Anderson PowerPole connectors, available from PowerWerx. The Ultra-Plug is lightweight and goof-proof, but the PowerPole is available in sizes designed from for 15A-350A, and is compatible with several splitters and adapters. Consider purchasing a few meters of silver solder, which contains about 2% silver, for very high current applications.

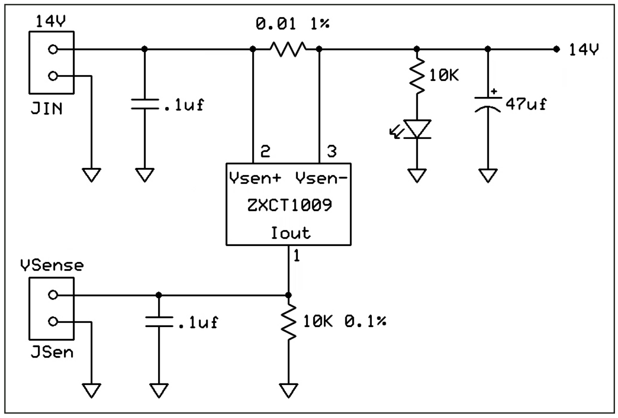
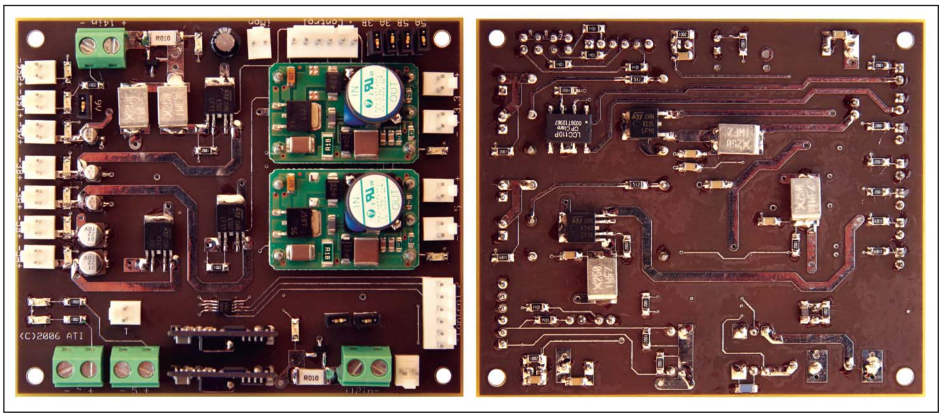


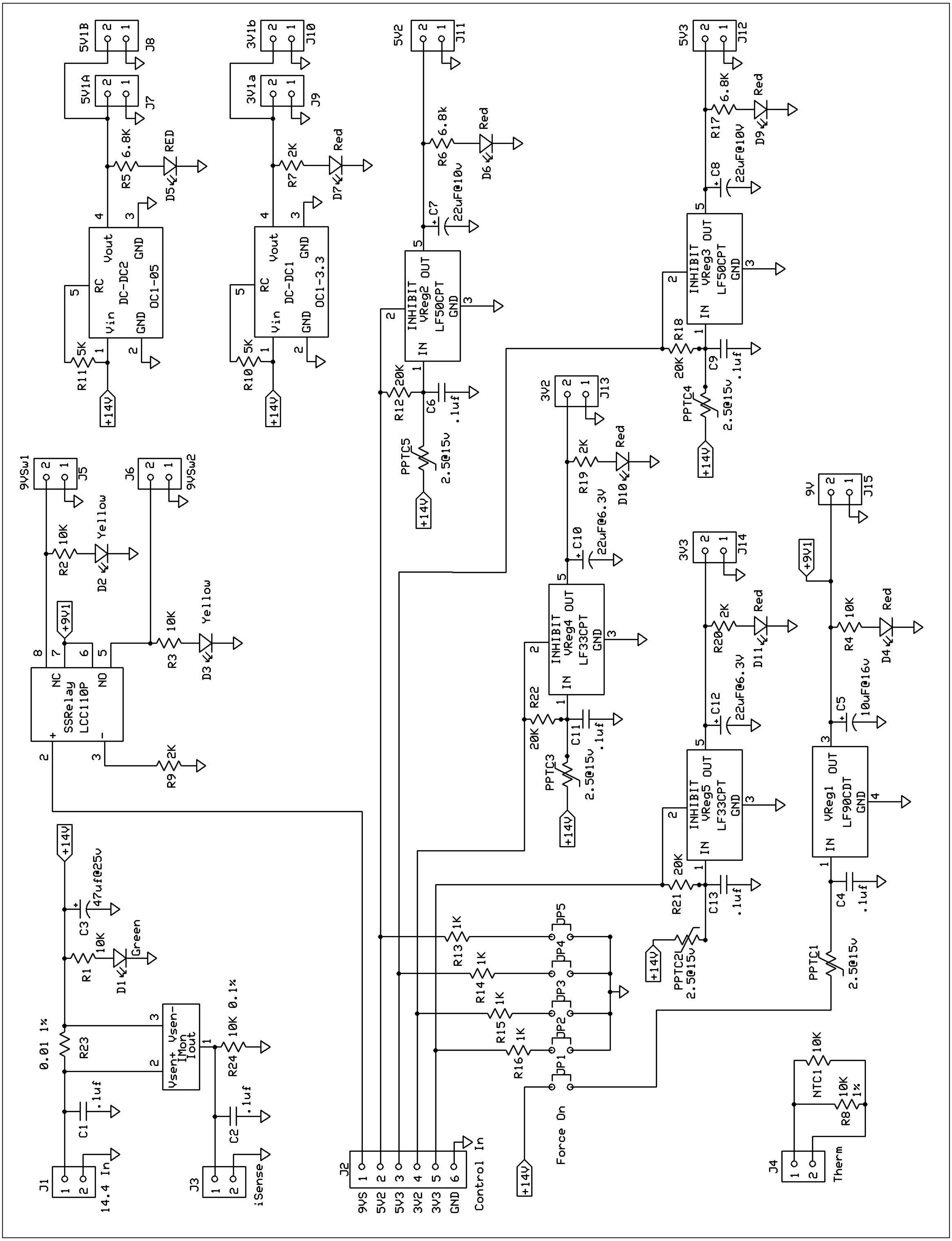
FIGURE 7. A low-resistance shunt resistor and a ZXCT1009 current monitor for real-time current monitoring.

Putting it Together

The multi-regulator circuit shown in Figures 9-11 is offered to readers who need a robotic power supply with multiple, isolated output voltages and switched subsystem capabilities. It is designed to support two batteries, a 14V-16V NiMH pack, such as the common 12-cell RC configuration used on the E-Maxx trucks, and a separate 12V source. The populated board measures 80 x 95 mm,

FIGURE 9. Top (left) and bottom (right) of the populated multi-regulator board.




J1
14.4 In
J3
iSense
J2
9VS
5V2
5V3
3V2
3V3
GND
Control In
J4
Therm
R23
0.01 1%
C1
.1uf
C2
.1uf
C3
47uf@25v
R1
10K
D1
Green
+14V
Vsen+ Vsen-
IMon
Iout
R24
10K 0.1%
R16 1K
R15 1K
R14 1K
R13 1K
DP1
DP2
DP3
DP4
DP5
Force On
NTC1
10K
R8
10K 1%
SSRelay
LCC110P
NC
NO
R9
2K
R2
10K
D2
Yellow
R3
10K
D3
Yellow
+9V1
J5
9VSw1
J6
9VSw2
DC-DC2
OC1-05
DC-DC1
OC1-3.3
Vin
Vout
RC
GND
R11
5K
R10
5K
R5
6.8K
D5
RED
R7
2K
D7
Red
J7
5V1A
J8
5V1B
J9
3V1a
J10
3V1b
VReg2
LF50CPT
INHIBIT
OUT
IN
R12
20K
C6
.1uf
PPTC5
2.5@15v
C7
22uF@10v
R6
6.8k
D6
Red
J11
5V2
VReg3
LF50CPT
R18
20K
C9
.1uf
PPTC4
2.5@15v
C8
22uF@10V
R17
6.8K
D9
Red
J12
5V3
VReg4
LF33CPT
R22
20K
C11
.1uf
PPTC3
2.5@15v
C10
22uF@6.3V
R19
2K
D10
Red
J13
3V2
VReg5
LF33CPT
R21
20K
C13
.1uf
PPTC2
2.5@15v
C12
22uF@6.3V
R20
2K
D11
Red
J14
3V3
VReg1
LF90CDT
C4
.1uf
PPTC1
2.5@15v
C5
10uF@16v
R4
10K
D4
Red
J15
9V

weighs 65 grams, and makes extensive use of surface-mount components.

The section of the supply shown in Figure 10 uses switching OC1 modules to provide 3.3V@1.5A and 5V@1.5A. Linear regulators and a solid-state relay are used to provide two 3.3V@0.5A outputs, two 5V@0.5A outputs, one 9V@0.5A output, and two switched 9V@0.5A outputs. The current available from the three 9V terminals is 0.5A total. The regulators in this part of the multi-regulator board are intended to work off of the same battery used by the drive mechanism.

The section of the supply shown in Figure 11 provides 5V@5A and 0.75-5.5V@5A outputs. It uses the SX-5A and assumes a 10V-14V source. Both sides feature current monitoring and LEDs on all outputs. The ExpressPC schematic and board design are available on the *SERVO* website (**www.servomagazine.com**) for readers who want to modify the board to their specific needs.

Does the multi-regulator fulfill the list of requirements established above for the perfect power source? Not quite. I'm still waiting for an inexpensive, lightweight, high-current module that works on any input voltage. Whether you opt to build the multi-regulator or use some of the technologies discussed here to design your own robot supply, you'll appreciate using one energy management system for all your robot designs. With power management issues resolved, you can focus on the more challenging aspects of robot development. **SV**

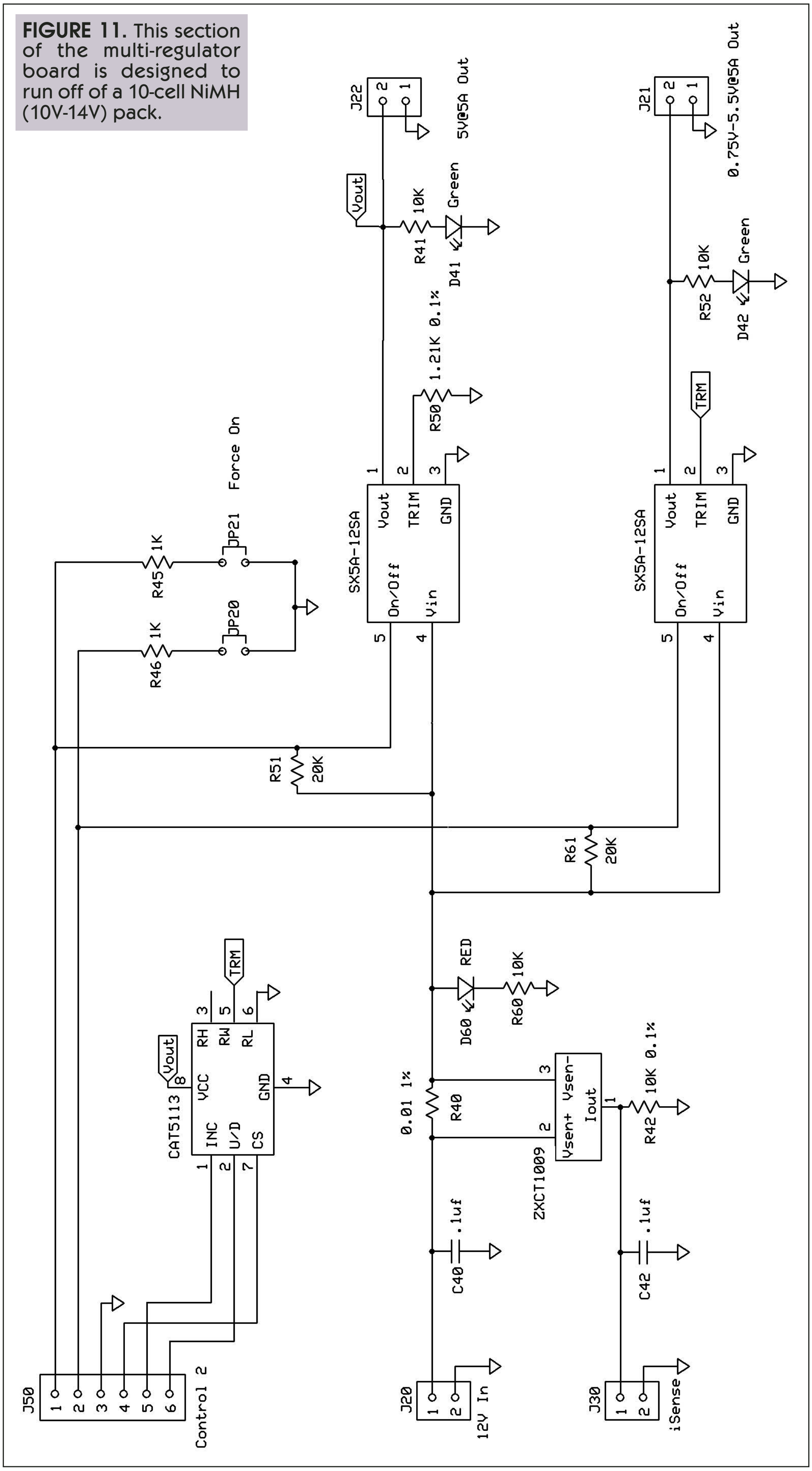


FIGURE 11. This section of the multi-regulator board is designed to run off of a 10-cell NiMH (10V-14V) pack.

RESOURCES

Mouser Electronics
www.mouser.com

PowerWerx
High-current connectors, splitters, and cables
www.powerwerx.com

Tower Hobbies
Connectors, cables, and silver solder
www.towerhobbies.com

Robotics Group, Inc.
RGi PowerCommander programmable power supply
www.roboticsgroup.com

Eagle Tree Systems
Micropower E-Logger
www.eagletreesystems.com/Micro Power/micro.htm

FIGURE 10 (left). Schematic of the mixed linear and switching regulator circuitry in the multi-regulator board. This section of the board is designed to run off of a 14V-16V, 12-cell NiMH pack or equivalent.

In Parts 1 and 2, we built the base and the controller for the FaceWalker robot. Now it's time to add the brain. If you remember back in Part 1 of the series, I told you that the face on the robot was one of the key components of the project. Without it, you have just another walker, albeit a cool one. We will be using a Pocket PC for the brain.

by Michael Simpson

FaceWalker

Part 3 — The Brain

Let's look at a few requirements:

- 300 MHz or faster processor, 500 MHz recommended
- Windows Mobile 2003 or 2005
- .Net Compact Framework 1.- SP3 or 2.0
- RS232 port
- 240 x 320 or better graphics
- ZeusPro Development Software

I understand that not everyone who wants to build the FaceWalker wants to use a Pocket PC. The good news is that the software that we create with the ZeusPro package will run on a normal PC as well as a Pocket PC. ZeusPro is a very easy and inexpensive way to do Windows development on the desktop and Pocket PC. It is perfect for robotics as it was developed for interfacing to the outside world. You can do all your development, testing, and debugging on the desktop and then simply create a desktop or Pocket PC executable.

For those Pocket PC purists, there is even a full-blown Pocket PC development environment that allows you to do all the development, debugging, and testing on the Pocket PC. The software portion of our brain is crucial, but we have a few finishing touches we need to make to our base first.

Finishing Up the Platform

If you have not done so already, you need to cut out your upper platform. This platform needs to be at least 7" in diameter, but can assume any shape you wish to add a different effect to the FaceWalker. I am going to assume you are using clear Lexan for your platform, but just about any smooth material will work. Most home centers sell 1/8" Lexan sheets and an 8" x 10" sheet will cost you under $5.

The Lexan sheet will come with a protective plastic

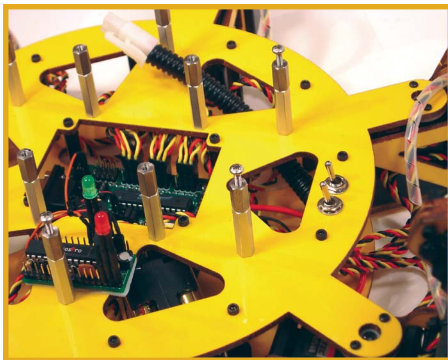


FIGURE 1

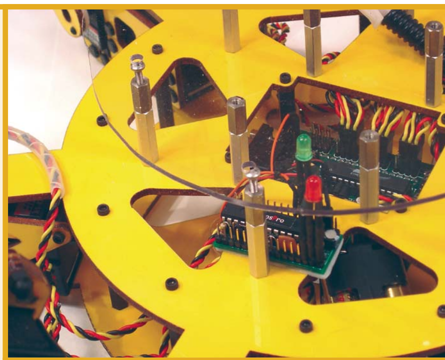


FIGURE 2

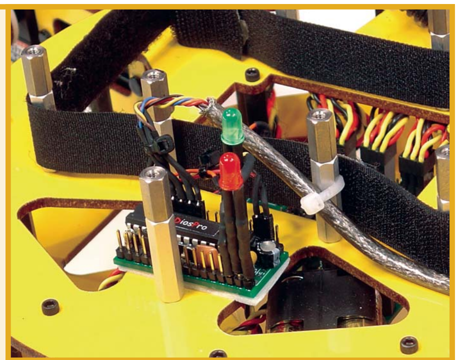


FIGURE 3

covering. Leave this in place while you cut out the platform. The covering makes a nice surface on which to draw your pattern and provides lubrication while you cut the sheet. If you have some scrap Lexan or plastic material without this covering, you will need to add some masking tape. As you cut the plastic, the covering shreds and provides a lubricant that keeps your scroll saw or band saw from melting the plastic. This will yield you a much better edge.

Once the platform shape has been cut out, you will need to mark out and drill the holes as outlined back in Part 1 of this series. We also need to create the holes for the two LEDs in our PS2 controller interface. This is done by inserting six #4 machine screws into the standoffs as shown in Figure 1.

You then place the platform on top of these screws, matching up the holes as shown in Figure 2. At this point, you can place a small mark on the Lexan over each LED. For standard size LEDs, a 7/32" drill bit is perfect but a 1/4" will do if you don't have that size. After this is done, you can paint the underside of the platform. Use a paint that was designed for Lexan. You can get this from your local hobby shop as it is what hobbyists use to paint RC car bodies.

In Part 2, you added a connector to a PS2 extension cable. It is time to once again connect that connector to the DiosPro. We need to take a small tie wrap and attach the cable to the standoff shown in Figure 3.

This provides some strain relief to the connector that we made back in Part 2. Place a small 1" piece of double-stick foam tape on the bottom of the PS2 connector and attach it to the platform as shown in Figure 4.

To attach the Pocket PC to the platform, we need a universal PDA mount like the one shown in Figure 5.

These have a flexible ball joint behind the head, and the neck can be positioned at any angle. They use a suction cup to hold the mount in place. I have had one of these in my car for years. The connection is quite strong and will last a long time as long as the surface is smooth and clean.

You can find one of these mounts for under $25 by searching for "Universal PDA Mount" on Amazon. You can also find them in some auto supply stores.

You are not restricted to this mount alone. The main head can be removed and mounted any number of ways. In truth, I mounted the head on a shorter cell phone mount on my first FaceWalker. I do recommend, however, that you use the cup for a while until you get your FaceWalker tricked out the way you want. This allows you to remove and reposition the mount as needed.

The actual PDA mount position is a matter of preference, but I placed mine as far back as it would go so that the Pocket PC would be somewhat centered as shown in Figure 6.

Pocket PC Serial Interface Cable

You are going to need a cable that connects your Pocket PC to the FaceWalker. Each and every Pocket PC is different and you will have to research your particular model in order to purchase the correct one. Most Pocket PC cables like the one shown in Figure 7

FIGURE 6

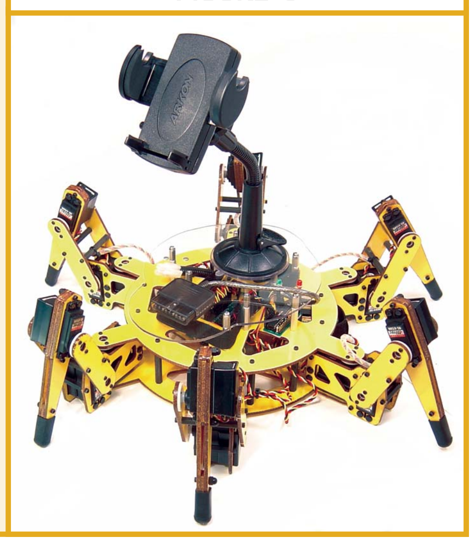


FIGURE 4

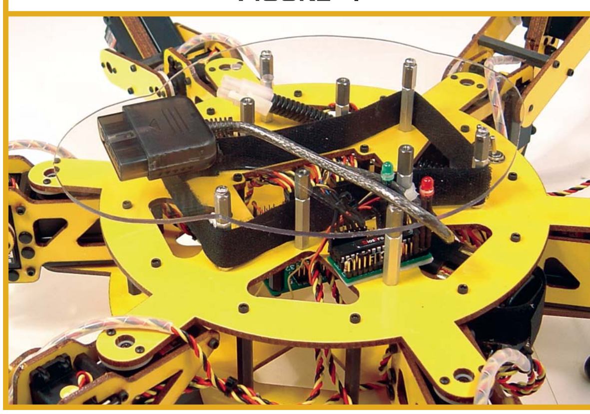


FIGURE 5

FIGURE 7

FIGURE 8

were designed to connect the PDA to the PC. This sets the Pocket PC as a DCE device and will not work when connected to the SSC-32. So, for most purchased cables you will need a gender changer and nine-pin null modem.

I attempted to use Bluetooth and a Bluetooth RS232 adapter and had little success. While Bluetooth has some good throughput speeds, I found that it has too much latency when used in an application such as this. Now, I'm not saying it can't be done. The other problem that Bluetooth has with operations like this is that if the Pocket PC gets turned off, you have to manually go back and reset all the connections, making it a pain to set up each time you change the batteries in the FaceWalker.

I also tried IRDA and Zigbee and neither panned out. With all the bad press that RS232 serial I/O gets, it provides a very reliable and rock solid interface in applications like this. If you decide to go with a cable like this, wrap the excess cable around the neck of the mount and route the cable through the inside of the base along the side of the SSC-32.

Make Your Own Pocket PC Cable

Another option is to build a cable. If you have an IPAQ Pocket PC, then you are in luck. The Kronos Robotics website has a step-by-step project showing you how to build the cable shown in Figure 8. The advantage to a custom-made cable is that it can be designed to the exact length and you won't need any gender changers or null modems. You can find the instructions on how to make one of these cables at **www.kronosrobotics.com/Zeus/IPAQcon.shtml**

Some Finishing Touches

Take a look at Figure 9. I added some split loom tubing to the serial cable and the PS2 connector cable. Walkers are notorious for destroying servo cables, so I have added thin plastic wrap to all of mine. You can also use thin split loom tubing for this, as well.

You may have also noticed in Figure 9 that I am using the shorter cell phone mount. You will still need the standard sized PDA head as the cell phone head is too small to hold a Pocket PC.

FIGURE 9

First Test

Before I get into the details of the software, let's test the completed FaceWalker from the PC. The FaceWalker desktop software requires you to have .Net installed on your computer. It will not run without it.

- Install the FaceWalkerDT_Setup software on your desktop computer.
- Place the FaceWalker on some sort of stand so that the legs are free to articulate.
- Connect a nine-pin serial cable between the FaceWallker SSC-32 board and the PC.
- Connect a wired or wireless controller to the FaceWalker PS2 connector.
- Power up the logic using the logic power switch.

At this point, the two LEDs should blink, then go green.

- Start the FaceWalker_DT application.

The red LED should start to flash. If it does not, you may need to change the com port as shown in Figure 10. The FaceWalker will wink continuously (nervous twitch) if it does not have a connection to the PS2 controller.

- Power up the servos using the servo power switch.

Move the right joystick forward slightly until the legs start to move. Once they do, let the joystick move back to the center position. The legs should stop moving. The more you move the joystick, the further the leg stride. Try all joystick directions.

FIGURE 10

Final Pocket PC Hookup

Now it's time to connect your Pocket PC to the FaceWalker.

- Install the FaceWalkerPPC_Setup software. You will need to have your Pocket PC attached to the PC when

you run this install.

- Once installed, disconnect the Pocket PC from your desktop and place it in the PDA mount on the FaceWalker.
- Connect the serial cable and any needed adapters to the Pocket PC and SSC-32 board.
- Connect a wired or wireless controller to the FaceWalker PS2 connector.
- Power-up the logic using the logic power switch.

At this point, the two LEDs should blink, then go green.

- Start the FaceWalker_PPC application on your Pocket PC.

The red LED should start to flash. If it does not, you may need to change the com port as shown in Figure 10.

- Power up the servos using the servo power switch.

Again, move the right joystick forward slightly until the legs start to move. Once they do, let the joystick move back to the center position. The legs should stop moving. The more you move the joystick, the further the leg stride. Try all joystick directions.

That's it!!!!! Construction is complete. Let's take a look at the software.

The Software

The FaceWalker brain software is broken down into four sections, which are: FaceWalker.txt, WalkerIKLib.txt, WalkerAnimationLib.txt, and Walker ActionLib.txt.

Let's take a look at each section in detail.

FaceWalker.txt

This is the entry point into the program. It contains the main loop and makes calls to various routines that allow the FaceWalker to operate.

When started, the program initializes various segments of the program and starts the com port. Once done, it enters a loop where all the processing in the program is taken care of.

- The first task in the loop is to check the menu to see if we need to change the com port.
- Next, we make a call to the animation routines to handle any face changes that were triggered.
- The program then calls the PS2 routines that pulse the DTR lead that signals the DiosPro chip to send us a reading taken from the PS2 controller. A set of six global variables called DualShock1-DualShock6 are populated with the current PS2 readings.
- After the buttons are checked and handled, a call to the IK routines are made to process the leg movements.
- The button settings are saved and we start all over and do it again.

WalkerIKLib.txt

This is the heart of the FaceWalker walking commands. IK stands for Inverse Kinetics. Here is what they do. Instead of using a state machine like many walker programs do, the position of each leg target is calculated based on the command you have given it. In other words, if you are telling the FaceWalker to spin, a mathematical calculation is made to calculate the new position of the leg from its current position.

What this all means is that you can give the FaceWalker the commands to move forward, sideways, and spin all at the same time and it will respond. It also means that from just about any position we can go to another position without having to return to a home position. This is not true of most state-based walker programs.

A big thank you goes to Laurent Gay for originally implementing the IK mathematics behind these walker routines. I simply ported the code to the Zeus language and encapsulated it into the ProcIK() subroutine.

WalkerAnimationLib.txt

This is an animation engine that handles all the sounds and face animations based on tiny lists that are controlled by a state machine.

High level commands like PlayHello, PlayHey, PlayGetback, PlayMean, and PlayCommandAttack are called to load a series of arrays called Eyecommands, Mouthcommands, and Sound commands.

These are then played back and choreographed based on the times loaded in their corresponding time arrays. The doface() subroutine is used at the gateway into this engine and is called on a regular basis from the main program loop.

WalkerActionLib.txt

The IK routines are used solely for making the FaceWalker walk. If we want to place the FaceWalker in a defensive stance, we need to make a call to the action library. The action library is where any action outside

Parts List

Kronos Robotics — **www.kronosrobotics.com**
- FaceWalkerDT install software
- FaceWalkerPPC install software
- FaceWalker source

- All the source code, as well as project updates, can be downloaded from the Kronos Robotics website at **www.kronosrobotics.com/Projects/FaceWalker.shtml**

KRMicros — **www.krmicros.com**
- ZeusPro: **www.krmicros.com/Development/ZeusPro/ZeusPro.htm**

Miscellaneous
- Double-stick Foam Tape. One 1" piece should do it. Any home center will carry this. A popular brand is 3M.
- Tie Wraps. You may need a couple of these to anchor some of the cables. You can pick these up at most home centers also.
- An 8" x 10" sheet of 1/8" Lexan. You can pick up cut sheets at the same home centers.
- Universal PDA Mount. Perform a search on **Amazon.com** for 'Universal PDA Mount.'
- Pocket PC serial cable. You will have to check with your particular Pocket PC manufacturer.
- Nine-pin Null Modem. You will need this if you purchase a serial cable designed to connect your Pocket PC to a desktop PC. Check **www.jameco.com** for this.
- Nine-pin Female-to-Female Gender Changer. You will need this if you purchase a serial cable designed to connect your Pocket PC to a desktop PC. Check **www.jameco.com** for this, as well.
- Split Loom Tubing. This is needed only if you want to hide or protect some of the cables. Most home centers carry this in the electrical section.

lift, the FaceWalker uses. By adjusting this value and slowing down the FaceWalker's speed, you could make him tip toe.

Triangle Button
- Slows down the overall servo speed.

X Button
- Speeds up the overall servo speed.

Square Button
- Face says "Hey."

Circle Button
- Face says "Hello."

R1 Button
- Attack Mode.

R2 Button
- Shy Mode.

L1 Button
- Cycle through commands. FaceWalker face will say the command.

L2 Button
- Do the command.

normal walking movement is done. This is done in a couple of ways.

One way is that we simply call a ComOutput command to place the servos in predetermined positions with various delays between the calls. The ACT_ATTACK command is an example of this type of action.

The other way is to use a set of low-level action segments. I have added commands like HipV, HipV, and Knee, as well as higher level commands like LegUp, LegDown, and LegFWD. By calling these commands, you can perform movements in loops. The Shy command is an example of this. You could use these commands to place FaceWalkers belly on the ground, then use the legs to create a wave.

The action library is also where we trigger face animations and sound to be played by the animation library.

FaceWalker Operation

Since we are using a PS2 controller, there are quite a few commands you can give the FaceWalker:

Right Analog Stick
- Left and right cause the FaceWalker to strafe left or right.
- Up and down cause the FaceWalker to move forward and backward.

The amount of movement is determined by how far you push the stick.

Left Analog Stick
- Left and right cause the FaceWalker to rotate.
- Up and Down set the height of the FaceWalker.

Select Button
- Locks the current height in place. Hitting it again releases it.

Digital KeyPad
- Left and right set the number of steps needed to make a leg movement. The lower the number, the faster the legs will get to their positions. For Pocket PCs, this is set to 4, and for desktop machines this is set to 8 at program startup.
- Up and down set the amount of leg

Many of these commands were created for the Robot Fest. Feel free to modify them for your own needs.

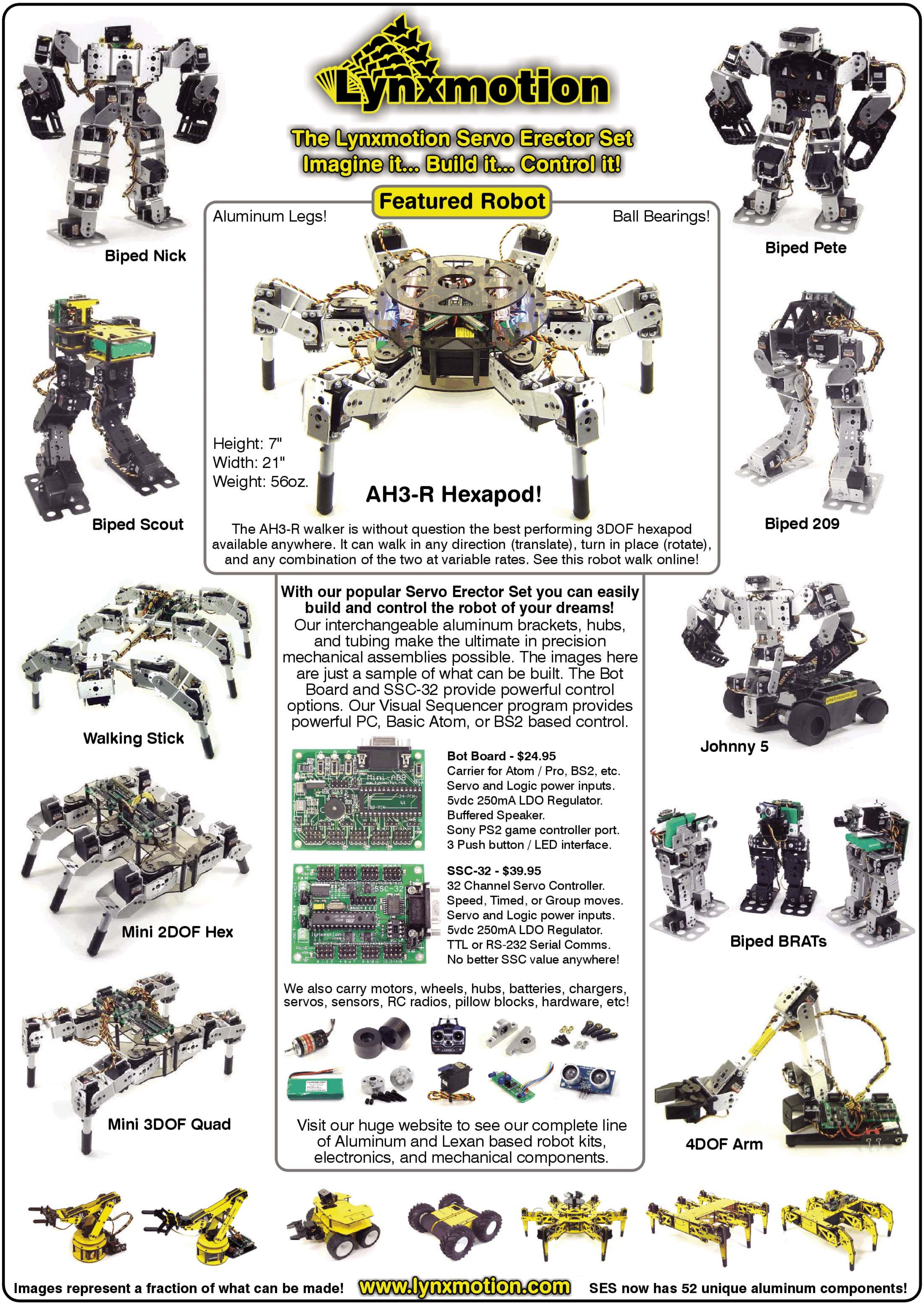
Final Thoughts

The FaceWalker has performed flawlessly. It has proven to become the attention-getter it was designed to be. I have even taken it to non-robotic events and it has never failed at stealing the show.

Going Further

The action and animation libraries were designed with expansion in mind. Take a look at the code and start playing around.

The DiosPro microcontroller's power has barely been tapped. It would be very easy to add some sensors to this interface such as a Sonar. Once done, you could start to add automation to your FaceWalker. It is absolutely feasible to make the FaceWalker totally automated. SV


Lynxmotion
The Lynxmotion Servo Erector Set
Imagine it... Build it... Control it!
Biped Nick
Biped Pete
Featured Robot
Aluminum Legs!
Ball Bearings!
Height: 7"
Width: 21"
Weight: 56oz.
AH3-R Hexapod!
The AH3-R walker is without question the best performing 3DOF hexapod available anywhere. It can walk in any direction (translate), turn in place (rotate), and any combination of the two at variable rates. See this robot walk online!
Biped Scout
Biped 209
With our popular Servo Erector Set you can easily build and control the robot of your dreams!
Our interchangeable aluminum brackets, hubs, and tubing make the ultimate in precision mechanical assemblies possible. The images here are just a sample of what can be built. The Bot Board and SSC-32 provide powerful control options. Our Visual Sequencer program provides powerful PC, Basic Atom, or BS2 based control.
Walking Stick
Johnny 5
Bot Board - $24.95
Carrier for Atom / Pro, BS2, etc.
Servo and Logic power inputs.
5vdc 250mA LDO Regulator.
Buffered Speaker.
Sony PS2 game controller port.
3 Push button / LED interface.
SSC-32 - $39.95
32 Channel Servo Controller.
Speed, Timed, or Group moves.
Servo and Logic power inputs.
5vdc 250mA LDO Regulator.
TTL or RS-232 Serial Comms.
No better SSC value anywhere!
Mini 2DOF Hex
Biped BRATs
We also carry motors, wheels, hubs, batteries, chargers, servos, sensors, RC radios, pillow blocks, hardware, etc!
Mini 3DOF Quad
4DOF Arm
Visit our huge website to see our complete line of Aluminum and Lexan based robot kits, electronics, and mechanical components.
Images represent a fraction of what can be made!
www.lynxmotion.com
SES now has 52 unique aluminum components!

by Dave Calkins

ROBOGAMES PREP:

RoboMagellan

I'm sure that you've heard of the DARPA Grand Challenge. Build a full-sized vehicle that can autonomously drive from outside of Los Angeles, CA to outside of Las Vegas, NV. Of course, even the most trimmed down Grand Challenge robots cost hundreds of thousands of dollars and the builders spent tens of thousands of hours making them.

But what if we could build mini Grand Challenge robots? Robots that were still autonomous and could go from one given GPS point to another, but could be built by the average robot hobbyist? That's what RoboMagellan is. A robot about the size of an R/C truck that has been reconfigured to autonomously drive around — operating outdoors rather than inside. But it's not just a matter of putting a BASIC Stamp on an R/C and collecting your trophy.

Ted Larson — two-time ROBOlympics medal winner with his RoboMagellan bot 'Odyssey' — points out that there are three key domains to master for building a successful RoboMagellan robot:

1) Mechanical design.
2) Electronics and sensor fusion.
3) Software — by far the biggest factor.

"My personal observation has been that most teams spend most of their time working on a good mechanical platform that drives around well, and the least amount of time on software. Our first year in Seattle, we did exactly this, and of course, like everyone else, we ran into something, and didn't make it to the final cone. Since then, we have been working on software and sensor upgrades, and finally have something that works well, two years later."

Many articles have graced the pages of *SERVO Magazine* covering this very problem. Odyssey has a main power distribution, motor control board, a sensor board, a GPS navigation board, a main CPU board for high-level decision making, and a camera board for talking to a CMUcam for cone finding. All the individual boards are tied together using an I^2C bus.

"Rusty" — Camp Peavy's RoboMagellan bot — is similarly controlled by five different BS2 Stamp computers. It is a network on wheels. One is the "Master Stamp" which coordinates the information from the four others. One Stamp is dedicated to getting GPS information; one is dedicated to getting compass

TAKE OUR ADVICE ...

Ted Larson offers some great advice for robot builders who want to compete at next year's ROBOGames:

- Build something now — Don't try to solve the whole problem all at once. I have seen some really elaborate designs that never made it off the drawing board. Better to have a robot that doesn't do everything correctly than to have no robot at all. Start simple, and work your way into a more complex solution.

- Get started building early — Don't wait until two weeks before the competition.

- Don't spend all your time on mechanical design — Good software is more than half the problem. Focus on software.

- Testing, testing, and more testing — Get yourself some road cones at a local safety supply store. If you can't make it work in your own backyard, then you certainly won't be able to make it work the day of the competition.

- Join the Seattle Robotics Society (SRC) Yahoo! Group or the Home Brew Robotics Club (HBRC) listserve. You will get great advice there from others who have working robots.

> “That’s what RoboMagellan is. A robot about the size of an R/C truck that has been reconfigured to autonomously drive around — operating outdoors rather than inside.”


Photo 1

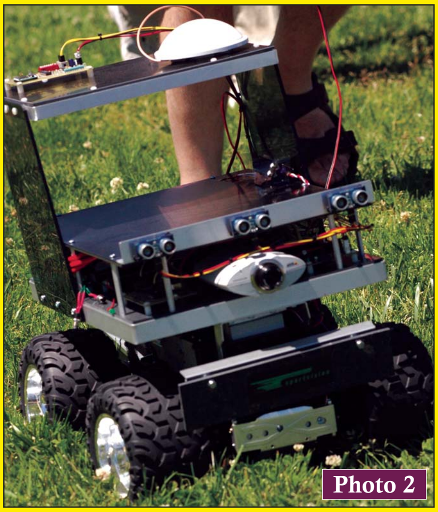


Photo 2


Photo 3

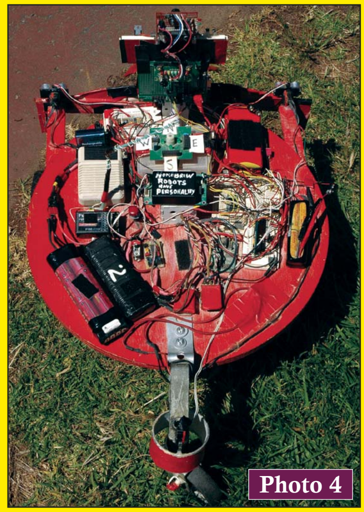


Photo 4

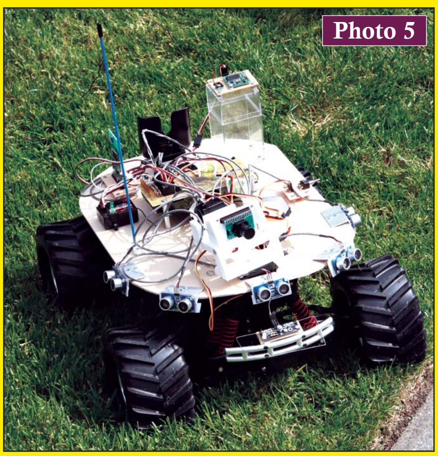


Photo 5


Photo 6


Photo 7

PHOTO 1. You’re almost there! A robot slowly makes it way to a bonus cone.

PHOTO 2. They may not be Stanley, but RoboMagellan robots can do a great job of finding their way along courses as well as their bigger cousins at DARPA can.

PHOTO 3. Getting a robot to a cone can be harder than you think.

PHOTO 4. Rusty — a bare-bones RoboMagellan bot — was built by Camp Peavy.

PHOTO 5. Marvin was a student-built robot that competed in 2005.

PHOTO 6. Ted Larson and Bob Allen’s robot “Odyssey” is custom-built from the ground up. It won the gold medal at ROBOGames 2006, and the silver the year before.

PHOTO 7. If just come as an onlooker, why not catch some sun while watching the robots?

information; one is dedicated to the ultrasonic array; and finally, one is dedicated to the CMU color camera. Each gathers its information in parallel and relays it to the Master Stamp which then uses that information in its decision making. Two robots completing the same task, yet with very different designs.

All of Odyssey's electronics do not help much without good sensory gear. Odyssey has five sonar units and navigates using a Garmin 18LVC GPS, combined with a self-designed six-degree-of-freedom inertial unit, and a two-axis magneto-inductive compass. It also has US Digital optical encoders mounted to the motors that get fed into the puzzle for dead-reckoning when necessary.

Ted and business partner Bob Allen custom-built a navigation board that takes all this data in, and smashes it out into simple vector-pursuit path planning algorithms. Once they have a path to drive, they feed the appropriate motor commands to a closed-loop control system on the motor control board, and then it's rubber to the ground to get there.

When Odyssey gets near where it thinks the cone should be, it starts looking at the camera for orange blobs. The CMUcam does this nicely. A recently acquired AVR-cam from JRobot will probably work even better. Once the bot finds the cone, it hones in to touch it using large, bug-like feelers sticking out in front of the robot. The feelers work well enough to know what's there, and to touch the cone without knocking it over.

Navigation Considerations

RoboMagellan is far more challenging than you might think. First and foremost, it's never a straight line from start to finish. So your software has to take into account how to go around an obstacle and get back on track to arrive at the destination (be that a bonus cone or the end point.) The terrain may be tricky to traverse — robots can easily get stuck or high-centered.

The course cannot be completed using GPS alone, which can be spotty from one point to the next. So builders have to plan on how their robot will respond when it's not getting a signal. Finding a cone from as close as a meter away is a more difficult problem than it initially appears to be, especially if you're using a CMUcam with its limited resolution. Bonus waypoints can be quite difficult to reach — they could be tucked away in a tight space, or as difficult as climbing a ramp or not falling off an embankment.

GPS reception on the course will be intermittent, so you need a backup navigation plan to get you through the dead spots. Haversine (Great Circle) math is difficult to do on an eight-bit MCU — if you build a RoboMagellan bot, plan on using a robust chip to do your math. Some competitors use hand-held GPS units that have the capability to do built-in routing. It saves you from having to do the GPS math on your CPU, plus it's easy to remove from the robot to collect waypoint data before you start a run.

It helps to have additional sensors that can be used to increase the accuracy of your GPS data. In RoboMagellan, GPS repeatability is

ROBOMAGELLAN RULES

- The robot must autonomously navigate to a "destination" point, and touch an orange road cone, in the shortest time possible.

- Several bonus waypoints (more orange cones) exist on the course which, if touched, give your final time a fractional multiplier, thus improving your time. Difficulty of each bonus waypoint determines the size of the fraction.

- Just about any type of obstacle (rocks, buildings, holes, etc.) can exist between the starting point and the destination. The destination will most likely not be visible from the starting line, nor will a direct path lead you to the destination.

- GPS coordinates of the start, finish, and bonus cones are handed out 30 minutes prior to the event, and contestants are allowed to walk the course and collect their own GPS data, if desired. You cannot pick up the robot and carry it around the course to collect GPS data, so your GPS unit needs to be detachable for taking measurements.

- The robot must fit in a 4 x 4 x 4 ft. square box, and cannot weigh more than 50 lbs.

- Course is laid out in a 300 sq. ft. area outdoors.

- Competition goes on rain or shine, so robot must be weatherproof. However, the robot will not be required to traverse a water obstacle to complete the course.

- Each robot is scored on the best of three tries to complete the course.

- Each robot is given 15 minutes to run the course on each of the three tries.

- Robot with the fastest, bonus-adjusted time to complete the course wins.

- Each robot must be equipped with either a wireless safety shutoff device, or a wired (tethered) dead-man switch.

- No other "remote" control beyond the safety stop measure is allowed.

PHOTO 8. Follow that human!

PHOTO 9. A crowd follows the first robot to compete at this year's ROBOGames.

more important than GPS accuracy. You should also use a compass and an inertial measurement unit to try to overcome GPS dead zones, and deliver more accurate GPS repeatability. While you might lose a GPS signal, you won't lose magnetic north — so you can follow your bearing until you get a reliable GPS signal again.

Finding the cone is not an easy task — don't underestimate how difficult it will be. Just because you can get the robot close, doesn't mean you will find it or even sense that it is there. As with all color matching programs, the lighting will change the apparent chrominance of the object — so be prepared for orange cones to look like another color. Lighting conditions are completely unpredictable, and determined by weather and shading conditions of surrounding obstacles.

Robot Platforms

One of the benefits of reading *SERVO* is that you're already ahead of the game! If you've been reading Chris Cooper's recent series "Mobility to the Maxx," you're already on route to build a great RoboMagellan bot. Everything Chris has described so far is a great start for your own RoboMagellan robot. It's got almost everything you need except a GPS receiver.

Of course, the Maxx is just one possible solution. You could use any number of robot platforms, including a legged walker. (I'd love to see a quick moving hexapod compete at RoboMagellan — there's no reason legged bots can't upstage the wheeled robots.)

RoboMagellan robots won't be cheap, however. Although Camp's "Rusty" was put together for just a few hundred, you could easily spend $1,000+ on a robot — but by using your ingenuity, I know that many builders could make a successful RoboMagellan bot for under $500.

RoboMagellan is a difficult, but rewarding event. Getting it right, however, takes a lot of time. Start now if you're going to compete. Find a few parks nearby your home and constantly practice with new coordinates and obstacles, changing parks frequently to get a feel for how your robot operates under different terrain, as well as better understanding just how difficult it can be to get accurate GPS data.

While building a successful RoboMagellan bot may seem like an insurmountable task, you've got nine months before ROBOGames 2007 to build a robot — so start now! Set deadlines! — Say, for example, by December, be able to make the robot move from point A to point B autonomously. By February, it should be able to do straight-line movement from two GPS points. The next month, add the cone-finding and touching abilities. Then, add a third cone as a bonus. Or, pick two GPS spots with an obstacle in the way, so the robot must go around it and find its own path again.

Building robots is never easy. This is a very hard challenge, but I'm sure many of you are up to it. If you start now, you just might be able to take Ted's place in next year's event! **SV**

FOR YOUR INFO

- **Name of Event:** RoboMagellan
- **Number of Robots per Event:** One
- **Length of Event:** 30 minutes
- **Robot Weight Range:** 50 lbs/22.68 Kg
- **Robot Dimensions:** 4' x 4' x 4'/121.92 cm^3
- **Arena Specifications:** N/A
- **Robot Control Specifications:** Autonomous

Go to **www.robogames.net/rules/magellan.shtml** for further information on this event.

Seasons Geekings

The monthly magazine for the next generation of robotics experimenters

Give your fellow Mech-Head a subscription to SERVO Magazine

Gift Subscription

Gift From:
Print Name ______________________
Address ______________________
City ______________________
State ________ Zip ______________
Phone ______________________
Email ______________________

Gift To: ❑ 1 Year ❑ 2 Year ❑ 3 Year
Print Name ______________________
Address ______________________
City ______________________
State ________ Zip ______________
Phone ______________________
Email ______________________

Gift To: ❑ 1 Year ❑ 2 Year ❑ 3 Year
Print Name ______________________
Address ______________________
City ______________________
State ________ Zip ______________
Phone ______________________
Email ______________________

U.S. Prices:
1 Yr $24.95
2 Yr $45.95
3 Yr $65.95

To order, photocopy this page and mail to:
SERVO Magazine
Po Box 15277
North Hollywood, CA 91615

❑ Payment Enclosed
U.S. dollars drawn on a U.S. bank only
❑ Bill Me *(U.S. Orders Only!)*
❑ Visa/MC/Amex/DiscoverCard

Card # ______________ Exp. Date
Name on card ______________

For faster service, subscribe by phone or online

877-525-2539 (toll free)
818-487-4545 (outside US)
www.servomagazine.com

When ordering, use promotion code Y6DXMS

An Interview With Tandy Trower

Manager of the Microsoft Robotics Group

by Phil Davis on behalf of *SERVO Magazine*

Microsoft is getting into the robotics business and a couple of months ago, I had the terrific opportunity to interview Tandy Trower who is the manager of the Microsoft Robotics Group. Following is a recap of our conversation together talking about the new Microsoft Robotics Studio.

SERVO: Tandy, tell us a little about yourself, what you've been doing at Microsoft, and what you are currently involved with.

TT: That could be a very long story, having been at Microsoft for over 24 years now. I have had a long career that included helping to start up a number of Microsoft initiatives in a variety of areas. For example, I was originally a product manager for our Basic programming products. Later that evolved into managing our full line of programming languages. From there, I transferred over to manage the first two releases of Microsoft Windows. Along the way I also managed the first version of Microsoft Flight Simulation and even hardware products like the Microsoft SoftCard (a Z-80 plug-in card for Apple II computers that enabled them to run CP/M) and even was an interim manager for what became our Office application suite for the Apple Macintosh. Having a keen interest in human-computer interaction I also founded the first usability labs, built prototypes of advanced user interfaces (including an interesting little technology called Microsoft Agent), and helped start the eHome division responsible for creating Windows Media Center.

I am currently the general manager of the Microsoft Robotics Group — a new product incubation developing an application development toolkit for the robotics community.

SERVO: Microsoft announced in June their entry into robotics software development with the Microsoft Robotics Studio product which you've been working on. Can you tell me a little about Microsoft's thought processes with respect to entering this specific industry?

TT: This was a direct response to the robotics community. About three years ago, I was serving as part of the strategic staff of our Chief Software Architect (a.k.a., Bill Gates). I found pioneers and leaders from the robotics community looking to engage with Microsoft, indicating that they felt that something significant was happening around robotics and looking for Microsoft to participate. And this same message came from the diverse parts of that community, including those using robots for educational purposes, universities doing robotics research, the hobbyist community, industrial automation companies, and the emerging consumer robotics sector. This led to a dialogue I started with Bill Gates, who asked me to go out and understand what was happening here in more depth.

So, after further time meeting with a variety of people throughout the community, I summarized my research into a proposal which also looked at what assets Microsoft might apply. Bill reviewed this and then asked me to review it with Craig Mundie, one of Microsoft's three chief technology officers, charged with investigating advanced technology strategies, and Rick Rashid, senior vice president of Microsoft Research. There was a unanimous consensus that now would be a good time to respond to the community and subsequently approved my proposal that we begin by creating a software development kit for that community that could help provide a good platform and catalyst to this emerging industry.

SERVO: So tell us, what is Microsoft Robotics Studio and what's included?

TT: Simply put, Microsoft Robotics Studio is an application development toolkit for creating software for robots. It includes three areas of software: 1) a concurrent, scalable, distributed runtime that provides a

consistent programming model across a wide range of robot hardware (from simple educational robots to industrial robots); 2) a set of development tools to make it easier to develop applications; these include a high-resolution 3D, physics-enabled simulation tool and a visual programming language; and 3) a set of sample services and code and tutorials (in a variety of different programming languages) that go from the basics of how to read a sensor to control a motor to how to support autonomous navigation.

SERVO: At what level is this product targeted at? By that I mean, are you targeting industry, university students, high school students, or the weekend hobbyist?

TT: The good news is that it is really targeted at all segments of the community. Like the early PC industry, robotics is a technology where the community has synergistic overlaps and that no single part of the community is sufficient to focus on.

SERVO: Do you really believe that the student and/or hobbyist can pick this tool up and become proficient after a short time?

TT: Yes definitely. We demonstrate this by including demos that illustrate how to apply this to robots as simple as LEGO Mindstorms or Parallax Boe-Bots. And to help this, we also provide sample source code and over 20 tutorials to help get them started. A number of universities are already looking to create curriculum programs around what we have.

The truly great thing is that the programming model is simple enough for novices, but powerful enough to apply to professional developers, meaning not only that it can be used by developers at a variety of skill levels, but also allow a growth path: The same fundamentals you learn for programming an educational robot can be applied to industrial robots.

SERVO: Okay. Tell me a little bit about the underlying architecture of the Microsoft Robotics Studio.

TT: I was very fortunate to find that Craig Mundie had been incubating some advanced technologies to address the fact that the programming model for the future is changing from the idea of monolithic applications to orchestration of many loosely coupled distributed processes. These technologies were the Concurrency and Coordination Runtime (CCR) and Distributed Software Services (DSS). While actually designed for advanced programming to address things like multi-core, multi-processor, and distributed processing models, I found these to be an excellent match to the needs of the robotics community.

SERVO: Would you mind going through each of those in turn: the Runtime, the Services Architecture, and the Concurrency and Coordination Runtime and tell us what it means to the average roboticist? That is, tell us in layman terms.

TT: The Runtime components are some of the most important features of what we are delivering. With any robot application scenario, you typically have to deal with the fact that you have many things happening at the same time. Traditionally, there have been two basic approaches to how you program that. The first is that you write a loop, which gathers your sensor inputs, does some processing interpreting the input, and then makes the appropriate output adjustments.

The problem with this approach is two-fold. First, while you are doing certain processing, you may be missing other valuable input coming in. In other words, while your code is commanding the wheels to move ahead, your sensors could be telling you that you are about to hit a wall, but since you are busy calculating your trajectory and adjusting the speed to the motors, you might not get this input in time.

The second problem is that this type of programming design is very brittle. One bug in your code and the entire robot application may crash. You also can't change the program while it is running. So the typical solution is to write threaded code, where you handle different processes on different threads. However, this proves to be a very complex and challenging programming task because of the timing dependencies and is usually only successfully done by very advanced developers.

What CCR does is provide you with a lightweight library of functions that takes care of this complexity for you, making things as simple as writing conventional single threaded programs. It determines when multiple threads are needed and manages them for you.

In addition, the underlying services oriented runtime (DSS) provides dynamic modularity. This means each process or service is isolated from the others, enabling processes to be stopped and restarted, or even replaced, while the rest of the system continues. In other words, if a sensor system malfunctions and you can shut it down or restart it, you don't have to reboot the entire robot. In this sense, the architecture better accommodates failure and promotes more robust operation.

Further, because the exposing state is a natural part of building a service, it is easy to inspect or change the state of a service simply by pointing a web browser at the service, then have it give you an XML representation of its state, as well as modify the state while the service is running. This approach is similar to what Sebastian Thrun of Stanford indicates was key to his success in his team's winning the last Darpa Grand Challenge — the ability to inspect and tweak things while the robot is running.

I could say a lot more, but I encourage your readers to check out our website at **www.microsoft.com/robotics** and read over the overview documents on the runtime or, better yet, check out our video on the Channel 9 website (which you can get to from our website).

SERVO: What kinds of robots have you hooked this up to already and what kinds of things do you have them doing?

An Interview With Tandy Trower

TT: We currently have our services architecture working with LEGO Mindstorms (RCX and NXT), Fischertechnik, Parallax, Robotics Connection Traxster, Lynxmotion Lynx-6 Robotic Arm, MobileRobots Pioneer P3DX, Coroware Surveyor, Kuka LBR3 Robotics Arm, and many more.

However, even if you don't have a robot, you can use our simulation tool to try out some of these robots. Can't afford a Pioneer P3 with a Sick Laser Range Finder or a Kuka robotic arm? No problem, there are models included to run in our simulation tool.

What you can do with them is limited only by the hardware, your programming ability, and your imagination, but note that we provide tutorials and samples in Visual Basic, C#, Javascript, and Python. And we have another programming option under development that if you can connect a couple of blocks together, you should be able to program a robot.

SERVO: I see that the LEGO NXT system is one of the ones you have tested. Can I actually go out and buy a LEGO System today, hook it up to the Microsoft Robotics Studio, and start making things happen?

TT: Absolutely. Robotics Tutorial 1 shows you (in C#, VB, or Python) how to read a simple NXT contact sensor. Robotics Tutorial 2 builds on that and teaches you how to use the sensor to turn a motor on and off. Things can progress from there.

SERVO: Are there other systems I can buy today which will work with your system?

TT: Because we document (and provide tutorials) on how to write a service, there should be no robot on the market that you couldn't use our platform with. Of course, it is easier if someone writes the basic hardware interface services for you. And, in addition to those we provide, other companies are creating services for their hardware.

SERVO: From experience with tinkering around with my own robotic projects, I have some specific questions I would love you to address. The first is in regards to sensor fusion. This is always difficult to do unless you are a wiz with Kalman algorithms.

TT: The services framework (DSS) I described earlier makes it much easier to do sensor fusion since you can easily write code that delivers messages between your sensor services and allows you to control when you process those inputs. For example, you can do "joins" on messages so your code only processes the input when all inputs are provided.

SERVO: What about PID loops? We always need to know how far out bots have traveled, or have the need to maintain constant speed when going up hills, etc.

TT: Again, the nature of the architecture makes it easier to program things like PIDs. However, we are also looking into what basic services like Kalman filters or PIDs we can provide that you could use, as well. For example, in the August version of our preview, we provide services that make it easy to integrate in GPS data, text-to-speech, and video capture. We hope to add to these basic services, but more importantly, because our architecture affords reusability of services, we hope and anticipate that the robotics community will find our platform a way to share interesting algorithms and solutions for common functions.

SERVO: How about odometry and world maps?

TT: Odometry is something you need some hardware support for. However, what's cool about our services architecture is that you create simple services that capture the data, feeding them to higher level services that do the odometry processing. Again, this means that you can design applications that are more portable between robots.

With regards to mapping, we don't currently provide any mapping services or formats. However, the distributed nature of the architecture means that it is easy to share maps between robots or between computers connected to robots.

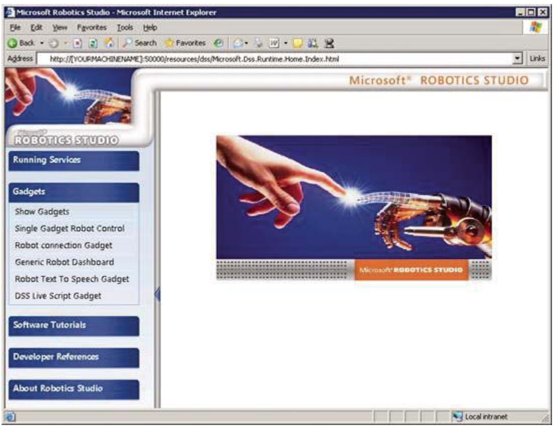
SERVO: Finally, from a specific point-of-view, it would be really great if we could get real-time telemetry from our bot as it wandered around. What about that?

TT: I covered this a bit earlier. The services infrastructure we provide means that if your robot has a connection to the network, you only need a web browser (doesn't even have to be Microsoft Internet Explorer) to check its state (telemetry). What's more is that you aren't limited to just a monitoring state, you can push back changes to the state. Also, I haven't mentioned that the services model isn't limited to the typical HTTP Get/Set model. We also provide for a publish/subscribe model which means that services can proactively communicate changes to their state to other services that would be interested instead of having to poll repeatedly.

SERVO: I understand the Microsoft Robotics Studio comes with a simulator and an integrated 'physics' model. Would you discuss this?

TT: Yes, this is another important feature of what we deliver. We believe simulation is important for so many reasons. It provides access to technologies one might not always have directly. For example, how many people can afford a Sick Laser Range Finder or a Kuka LBR3 robotics arm? But having access to these in our simulation tool means everyone can experience what this is like to program and use these. Of course, simulation can also be used to streamline the development process since you can re-run your code against the simulation robot and discover whether your code causes the robot to crash into the wall, without destroying your hardware. It also means though you could design robots that don't yet exist and try them out, even write code for them, before you build the physical hardware.

An Interview With Tandy Trower



SERVO: How well does the simulator work? I mean, can I develop my robot application using the simulator and have it transfer over accurately to the real-world environment?

TT: For the most part, the answer is yes, since we have integrated in the Ageia PhysX physics engine which provides emulation of many aspects of how objects interact in the real world. Ageia has been a leader in providing this kind of realism in some of the most advanced computer games on the market. They even have a hardware plug-in card which effectively gives you a physics co-processor for your PC.

However, there are obvious limitations. The real world is full of noise (inherent unpredictabilities). The physics simulation being a calculated reality doesn't necessarily deliver the same kind of variations, so you might have to add that (which you could as a service) to get more realism. In addition, there are a number of effects in the real world that may be difficult to model. Further, modeling things to behave as they do in the real world can sometimes take time.

So, simulation may not be a perfect substitute for real-world testing, but it is a good tool to have. Also, the distributed nature of this service means it is easy to allow others to interact with you in the simulation. So I can imagine all types of interesting collaborative and competitive scenarios between robots using our simulation tool.

SERVO: Tandy, what else would you like to add which you think *SERVO* readers would be interested in knowing?

TT: I would like to mention the openness of what we are doing. We have worked to make our platform easy for others to contribute to by documenting the services interfaces and including tutorials and sample sources. By doing this, we hope to provide to the community something anyone can contribute to and overall expand the accessibility of robot technology to a larger audience.

SERVO: Where do you see all of this heading, both from a Microsoft point-of-view and for robotics in general?

TT: I personally believe that robotics is the next major evolution of PC technology. It is incredible how the robotics industry follows the same trends that the PC industry did. In the late 1970s at the birth of the PC industry, there were initially no hardware or software standards, development was hard, and people asked why you would own a PC. The same is true for where robotics is today. Similarly, the same energy, anticipation, and interest across a diverse community that was present in the early PC industry can be found in the robotics community.

Consider, too, the similar rate of progress. When I was a kid, when I built robots, I had to pretend they were robots, but my kids grew up tinkering with LEGO Mindstorms and Aibos. Just eight years ago, when LEGO introduced Mindstorms, it was based on an eight-bit processor and IR communication. Their new NXT is 32-bit technology with BlueTooth.

Today, our PCs respond to us mostly through the keyboard and the mouse, yet more and more they are equipped with other sensors such as microphones and cameras, primarily motivated to support person-to-person communication, but it is easy to see that as these sensors become pervasive and technology to use them becomes more advanced, that your PC will use them to sense the environment and interact with you in a richer way.

Further, I believe that distribution of processing is key for robotics technology to succeed. Already our digital lifestyles are increasingly dependent on interaction between technologies, whether it is downloading music from websites or reading our email on our cell phones. Robots must participate in this digital ecosystem, as well, and therefore must be connected. And being connected allows for richer scenarios.

And that is the one question you didn't ask me, which is what do I think are the key scenarios for robots. Maybe it is good you didn't because I don't have a good answer for that yet. It is as hard to predict as it was for PCs in the '70s, and for them, recipe management didn't turn out to be the driving application.

However, I will offer that I think remote presence scenarios — where the robot enables you to be somewhere else — may be one of the most important scenarios that I expect to see robots offer. You can already see this in use with the military, but also with doctors making their patient rounds remotely piloting a robot.

We may still be 5-10 years off before robotics hits the tipping point, but it seems as inevitable as was the anticipation that PCs would be pervasive. Remember, we are 30 years into the PC evolution, so another 5-10 years is not that far off, and the signs that it is coming already surround us in our cars, appliances, and other forms of smart technology.

SERVO: If someone wanted to get started using the Microsoft Robotics Studio, how would they go about it?

TT: Best way to get started is to download it from our website at **www.microsoft.com/robotics**. As mentioned, we include a number of tutorials and sample code to help get you started. Thanks for the time to share what we are doing with your readers. We welcome them to try it out and let us know what they think. **SV**

2006 RFL Nationals

Photos by Sam Kronick, Michael Maudlin, and Pete Smith.

by Pete Smith and Charles Guan

This year's Robot Fighting League's (www.botleague.net) National Championships were held on August 11th in Minneapolis, MN.

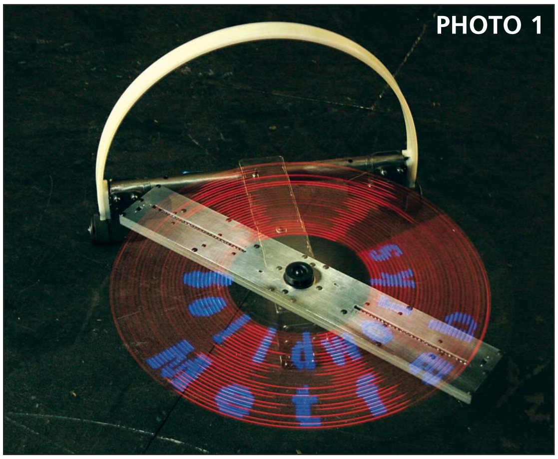


PHOTO 1

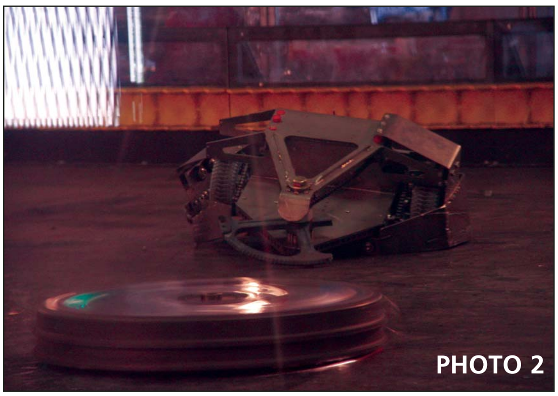


PHOTO 2

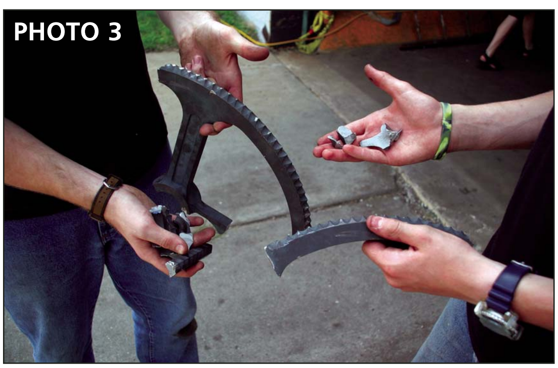


PHOTO 3


PHOTO 4

The Orange Terror alert and ensuing chaos at the nation's airports caused problems for some entrants. Team Ti Joe and their lightweight "Joe 2.0" arrived late and Team Wazio and the number two ranked (see **www.botrank.com)** 12-lb Bot "Wedgio" missed the contest entirely. Team Logicom had other air travel problems when Brian Nave's laptop was stolen out of his checked luggage while en-route (luckily, the criminals did not know the value of the bot parts that were not taken!).

The late choice of venue and mid-west location may also have reduced the bot numbers in some classes, but there were strong fields in the 12 lb, lightweight, and especially in the 30 lb Featherweight class where three out of four of the top ranked bots were among the entries. Team Moon and Team Rolling Thunder — both from North Carolina — made an especially strong effort bringing a total of eight bots to the event including two super heavyweights!

The 12 lb up to Super Heavyweight fights were held in the large 40 ft x 40 ft arena with its massive steel bumpers and 1" thick spaced polycarbonate walls. The 3 lb beetles and 1 lb ant battles were in the small arena that Team Killerbotics had brought on a trailer from Wisconsin. The big arena is superb with a reasonably smooth steel floor and competitor-operated heavy hammers in each corner which were used in the lightweight and above fights. The only possible criticism would be that the lighting used caused a lot of glare for the competitors and was not really bright enough for good photography. The small arena was perhaps a bit tight for the beetles, but worked well enough with its excellent lighting and visibility.

The venue opened up on Friday early to allow safety testing and gave some teams a chance to reassemble and test their bots after shipment. Robert Woodhead of Team Mad Overlord tried out a "persistence of vision" blade on his 30 lb "Totally Offensive" entry (Photo 1) which uses two arrays of red and blue LEDS to display writing and simple graphics. The system is not yet tough enough to use in combat, but he's working on it.

The main competition got started by 11:30 AM on Saturday. The weight classes with only a few entrants used a round-robin system where each bot had to fight every other entry until there was a clear winner. Those classes with enough bots, i.e., the 12 lb, 30 lb, and Lightweights ran the usual modified double elimination brackets (each bot has to lose twice to be eliminated, except for in the final fight where it is sudden death). It was a very relaxed event where everybody was given plenty of time for repairs and recharging, and only

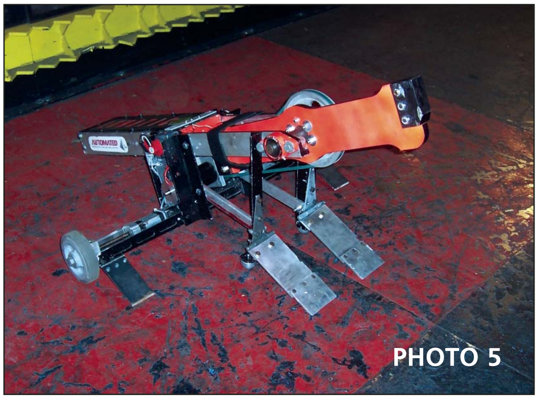


PHOTO 5


PHOTO 6


PHOTO 8


PHOTO 7

terminally damaged bots had to forfeit fights.

It quickly became apparent that it is very easy to over-harden (by accident or design) the S7 tool steel many use for weapon blades or teeth. A beautiful solid S7 ring on Team Moon's heavyweight "Eugene" shattered completely in an encounter with Team Logicoms ever powerful "Shrederator" (see Photos 2 and 3). Totally Offensive and Relic also suffered from breaking teeth in fights with Xhilarating impaX 2.0 and Killabyte.

The Super Heavyweights class was won by Team Kontrolled Kaos with "Psychotic Reaction" who fended off the frenzied attacks of Team Moon's "Starhawk" (Photo 4) who had to settle for second. Brian Nave's "Shrederator" dominated the Heavyweights with "Eugene" taking the runner-up prize.

Team Toad's "Ice Cube" took first place with newcomer "Lunatic" (Photo 5) coming in second in the Middleweights.

The 60 lb Lightweights field was won by Team Hawg's Blade spinner "Son of Whacky Compass" narrowly beating Team Killerbotics flame throwing "Goosfraba."

The 30s were probably the toughest fought weight class. Excellent newcomer "Relic," two-time Nationals winner "Totally Offensive," and a tougher than ever "Killabyte" made this a very tough group. Epic battles showered brilliant white titanium sparks in all directions (Photo 6) until only "Killabyte" and "Xhilarating impaX 2.0" (driven by 12-year-old Andrew Smith of Team Rolling Thunder); (Photo 7) were left fighting it out. A badly damaged "Xhilarating impaX" grabbed the victory as one last big hit broke his opponent's radio receiver.

The highlights of the 12 lb class were attempts at self-immolation by flame throwing "Kitt" and a father-son finals battle where Pete Smith of Team Rolling Thunder driving "CheepShot 3.0" (Photo 8) narrowly beat bar spinner "Surgical Strike" driven by his son, Andrew.

In the beetles, "Itsa?" beat out "Nuclear Kitten" for first place and in the Ants, "UnderWhere?!" took first with "Pop Quiz" second.

Many thanks to everyone at the Midwest Robotics League (**www.midwestroboticsleague.org**), Mechwars (**www.tcmechwars.com**), and many others for putting on an excellent event! SV

ROBOTICS RESOURCES

Tune in each month for a heads-up on where to get all of your "robotics resources" for the best prices!

BY GORDON MCCOMB

Taking Stock of Robotic Tanks

Since World War I, the tank has become the symbol of military battles. The hulking mass of iron plowing over the Earth, the shrieking sound of metal crushing against the ground, the blasts of fire from its cannon, all combine a sense of awe and respect. Little wonder that the tank design is popular among robot builders. The same principles that make a military tank superior for uneven terrain apply to robots, or in fact, any other type of treaded vehicle.

A number of robots in science fiction films run on tracks. There's Number Five from the movie *Short Circuit*, Robot B-9 from *Lost in Space*, and many others. Unlike their walking cousins, these robots actually look feasible. Their wide tracks provide a solid footing over the ground.

In this installment of Robotics Resources, we'll look at robots that use tank treads for locomotion. We'll cover some design basics, then discuss where to find suitable treads for your next 'bot creation.

Getting Tracked Vehicles to Move and Steer

All tanks share a similar design. Two long, chain-like tracks — also called treads — are mounted parallel to each side of the vehicle. A separate motor powers the tracks in either direction via a sprocket; the toothed design of the sprocket ensures that the drive mechanism doesn't just spin if the track gets jammed. Though there are some hybrid vehicles that use treads in addition to wheels, in the typical tank, locomotion is provided solely by the two tracks. The tracks are kept parallel to the vehicle by the use of idler rollers placed along the sides.

To move forward, both tracks move in the same direction. To reverse, the tracks move in the opposite direction. To turn, one track moves forward, while the other moves backward. This is a form of *differential steering* — steering is accomplished by the difference in the motion of the tracks relative to one another.

Because a tank track exposes a considerable amount of its surface onto the ground at any one time, in a turn the tracks must actually slip, or skid, over the earth. For this reason, tanks are often said to have *skid steering*. The part of the track furthest from the mid-point of the tank skids the most. For the typical military tank, tractor, or construction vehicle that uses tracks, this poses no serious problem. These vehicles are primarily designed to operate over dirt, and the treads are hard metal. Both track and surface have a relatively low *compliance*, which can be described as the resiliency, or "give," of the materials.

Turn now to the typical small robotic tank. These also use tracks mounted parallel to the body of the vehicle. These also turn by operating the tracks in opposite directions. Because of size, cost, and weight concerns, the track material on most robot tanks is rubber. Rubber has a higher compliance than metal, and if the robot is operated over a surface that is also fairly compliant, turning may be difficult for the little tank. The contact of two compliant materials limits the ability to skid in turns. For this reason, it's not uncommon to limit the operation of a robot tank to certain surface conditions only. Dirt, concrete, even low-nap carpet are generally preferred over thick carpet, for example.

Another potential issue of using a rubber tread is static friction, or so-called stiction. (There may be other frictional components involved, but we'll bypass them for this simplified discussion.) With stiction, a rubber tread may have difficulty skidding over a highly polished material, such as a glass tabletop or some hardwood floors.

There are numerous techniques to reduce the steering problems inherent in all treaded vehicles. One is to use a less compliant tread material. Not all rubber compounds are equally elastic. A good rubber tread for a tank design exhibits only limited elasticity (stretch). The surface of the rubber is smooth, and may have molded-in "cleats" that reduce the surface area of rubber touching the ground at any one time. With less surface area, there is less rubber to skid.

Another approach is to use a tread made of something other than rubber. Hard plastic is a valid alternative to metal, and plastic treads are found in a number of hobby robot products. The

VEX Robotics Tank Tread Kit, at about $30, is an excellent example of a effective tread system made out of plastic. The kit — which is designed for the VEX line of robots but can be adapted to other applications — consists of a series of links that you put together. The track is powered by a sprocket connected to a motor; idler rollers keep the tread aligned onto the robot base.

Another example of hard plastic tracks come from an outfit called Johnny Robot (named, no doubt, after the robot character of Short Circuit, which as noted earlier runs around on two tracks). The tracks are also available from a number of resellers, including Budget Robotics (my small firm), Pololu, and Robotics Connection. These tracks are composed of ABS plastic links, connected by stainless steel. You connect the links together to make a track any size you want. Plastic sprockets and idlers are also available to make a complete tracked system.

If there is a disadvantage to hard plastic tracks it's that the plastic may slip over hard surfaces — the exact opposite of rubber treads. Depending on the design of the track, you may be able to overcome this by applying small pieces of rubber material over the track segments. This provides enough compliancy to improve locomotion and steering. Lynxmotion recently added their own design of track that uses hard plastic, but also incorporates a rubber pad for added traction over smooth surfaces.

Rubber and plastic (or metal, for that matter) tracks also differ in their resistance to detracking — also called derailing, or throwing a track. Detracking occurs mostly when negotiating a turn. This is when the frictional forces acting against the track are at their highest. As the vehicle attempts to turn, heavy sideways pressure is exerted at the front and back of the track. If the pressure is great enough, the track may come off its drive sprocket or guide rollers.

Detracking is the most problematic with highly elastic rubber treads. The more elastic the track material, the more readily it will stretch during a turn. The problem is magnified if the tank is loaded down with weight. The heavier the vehicle, the more likely you'll have a thrown track. To limit this problem, do the following:

- Reduce the weight on the vehicle.

- Make slower turns.

- Try to find a rubber tread that doesn't stretch as much. The lower the elasticity, the less likely the tread will pop off.

- As necessary, tighten the track by adjusting the distance between the drive sprocket on one end, and the idler roller on the opposite end. This limits the track from stretching too much more. Avoid over-tightening, which can deform the tread and place excessive stress on the drive components.

- Decrease the surface area of the tread on the ground. You may do this by changing the elevation of the idlers toward the front and back.

- Experiment with the width between the tracks. Longer, narrower track widths resist turning more than shorter, fatter widths.

- Add "keepers" to the idlers that don't touch the ground. The keepers are like oversized rims that keep the track in place.

By their nature, plastic tracks don't stretch, so assuming they are placed snugly onto the sprocket and idlers, detracking is rare.

FIGURE 1. A close-up of a drive sprocket and corresponding cogs on a tank tread. The sprocket positively engages into the cogs to prevent the drive from simply spinning if the track gets stuck.

Finding Sources for Tracks

With the rising popularity of tank designs for robotics, more and more online retailers are offering track solutions. See the Sources section for a more complete list. You can choose from among ready-made tank-style bases that come complete with tracks, or you can purchase just the tracks.

Note that each type of track has a different method of engaging with its drive sprocket (see Figure 1), so if possible, always purchase a track with its corresponding sprockets and idlers. Otherwise, you will be left having to fashion something yourself, or settling for a system that doesn't work as well as it should.

One of the best sources for inexpensive rubber tracks is toy tanks. These are sold in different scales, from about 1:64 (miniature) to upwards of 1:10 or even 1:6. (The scale is the ratio of the size of the model to its original. A scale of 1:24, for example, means the model is 1/24 the size of the original. Most toy tanks are in the range of 1:24 to 1:32 scale, and these sizes are ideal for small amateur robots.)

Look for a toy where the track is not too elastic, and where at a minimum, the drive sprocket and idler rollers can be removed and placed on your own custom base. Some toy tanks offer easier hacking, where you can simply remove the turret and top of the vehicle, and replace the electronics with your own microcontroller and H-bridge. You do not need to build a body for your robot. The Sources section lists a few notable finds.

Perhaps the most often used track

is made by Tamiya, and sold by itself as item number 70100, Tamiya Track and Wheel Set. A number of online sources, such as Tower Hobbies (**www.towerhobbies.com**) and Hobbylinc (**www.hobbylinc.com**), offer this track. The track is also included in a few other Tamiya products, as the Tamiya Tracked Vehicle Chassis Kit and the Tamiya Remote Control Bulldozer Kit. These come with one or two motors, respectively.

The Tamiya track is rubber, and comes in segments of various lengths. You put the segments together to build a track. The segments connect using a little nub on the edges of the track. Despite how it sounds — or even looks — the tracks are actually quite robust, and seldom break apart unless forced. In a pinch, you can glue the pieces together with a flexible adhesive, such as silicone caulk. Make sure the adhesive doesn't seep into the part of the track that engages with the sprocket, and that the seam is smooth. Sprocket and idler rollers are included, and you pick out the parts you need.

The same Tamiya track is used in robot bases sold by Rogue Robotics, Parallax, Budget Robotics, and several others. These bases are custom designs that make use of the Tamiya track, drive sprocket, and idlers. The Parallax offering is a retrofit for their Boe-Bot robot chassis. It turns the Boe-Bot, which ordinarily comes with wheels, into a tracked vehicle.

If you're looking to construct a large robot, say about two feet long or more, you'll want to look for a 1:12 to 1:6 scale tank. These are occasionally available at the larger discount retailers, such as Wal-Mart. The fall and Christmas seasons are ideal times to find these. The toys are physically large, and stores may not stock them year round because of space considerations.

A relatively new source of tanks in this size range are intended for BB and paintball play, and you can find them at online retailers specializing in air guns for these sports. Prices vary from about $50 to over $200, depending on the size, brand, and quality of the tank. All the tanks are remotely controlled. Note that quality varies greatly in this genre, with users sometimes reporting "DOA" products when they receive them through the mail. Make sure to buy yours from a reputable dealer who offers free shipping for the return of defective merchandise. As these are quite large and heavy, shipping can get expensive.

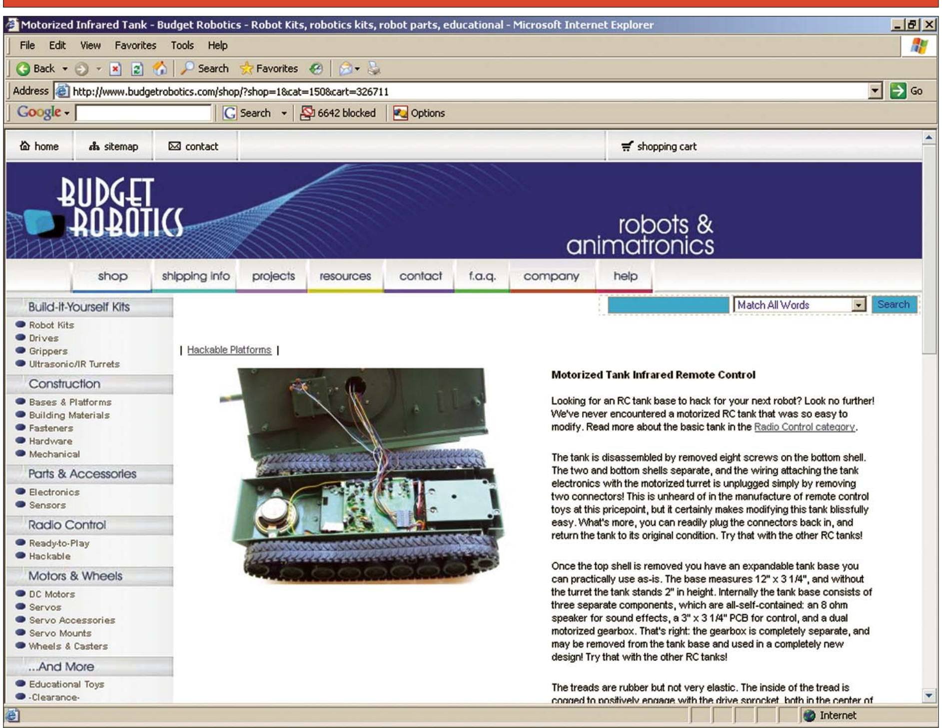
For smaller and more precision tracks, plastic scale models are another source worth looking into. However, be aware of possible sticker shock! Though there are some under $100 models with plastic or rubber treads you can reuse for your robot, the better quality treads are found on kits costing upwards of $500, $1,000, even $2,000. Tamiya, as an example, is a manufacturer that offers plastic tank kits at all these price points.

Automotive and machine timing belts are another source of tracks. Don't forget the matching sprockets for these belts. The typical timing belt has a cog on one side only. This cog engages with the drive sprocket. Some timing belts are double-sided, and have cogs both inside and outside. These are often used in serpentine arrangements where the front and the back of the belt engage with various mechanical parts. Double-sided timing belts are preferred for use as tank tracks, because they have a cog on the inside for engaging with the drive sprocket, and a cog on the outside that act as cleats for enhanced locomotion.

You can find timing belts at automotive shops, junk yards, and industrial supply outlets. They come in various lengths and thicknesses. Most are fiberglass-reinforced rubber, which means they are flexible, but they do not stretch.

Finally, if you have your heart set on metal treads, check out the tracks for snowmobiles. Most tracks are sold for a specific make and model of snowmobile (use an online search for 'snowmobile tracks'), but you will find they are available in all types of lengths and widths. Expect to pay good money for a set of snowmobile tracks.

Toys are a good source of tank treads and parts. This hackable tank at Budget Robotics is easily modified to a robot base.



Sources

Google.com or Yahoo.com search: "airsoft tank"

Use this generic search to locate

any of several dozen online resellers of remote control tanks for the air-powered BB and paintball sports. eBay is another option. Make note that a 1:16 scale tank is approximately two feet long. Features such as sound effects, smoking effects, and so forth are less critical than the overall quality of the tank. Be sure to look for and read user comments.

Acroname
www.acroname.com

Reseller of the Johnny Robot tracks and sprockets. The tracks are sold in sets of 20 segments and 20 links. Depending on the length of the track you want, you'll probably need two sets to build a tracked vehicle. They sell some alternative colors (at least yellow, when I checked) if you're not wanting the basic black.

Budget Robotics
www.budgetrobotics.com

Budget Robotics (my company) offers a number of tank-style robotics and track solutions, including ready-made robot bases. Sells Tamiya track kits, as well as kits with more robust custom rubber treads. Budget Robotics resells the Johnny Robot hard plastic tracks in a "Track Drive" kit, which allow development of a customized tank robot. Also offers some hackable toy tanks.

Johnny Robot
www.johnnyrobot.com

Makers of a unique hard plastic track system. The tracks are individual segments, connected together with a stainless steel pin. Tracks of any length are build by putting the required number of segments together. The company sells directly in quantity, and offers the product through distributors listed on their Web page.

Lynxmotion
www.lynxmotion.com

Manufactures and sells a unique adjustable-length track system. The links are made of 2" wide hard plastic, and are capped off with rubber pads. The company also sells a robot kit based on the tracks.

Parallax
www.parallax.com

Look for their Boe-Bot Tank Tread Kit, which turns your Boe-Bot robot into a tank. The treads, sprockets, and idlers are from the Tamiya Track and Wheel kit.

Pololu
www.pololu.com

Reseller of the Johnny Robot tracks and sprockets. They sell the track segments individually, or as a set of 40. Sprockets are available for Futaba-compatible servos, as well as small DC motors sold by **Solarbotics.com** and others.

RC Armory
www.rcarmory.com

RC Armory is a manufacturer and seller of large (1:8 scale) battle tanks with metal tracks. They offer their wares in kit form and various options. Rather than purchase a full tank and tear out what you need, you can get just the parts — such as the drive system — and save money. Prices for the drive unit, without electronics, start at about $500.

Robotics Connection
www.roboticsconnection.com

Offers a treaded vehicle based on the Johnny Robot track segments.

Rogue Robotics
www.roguerobotics.com

Robot base built from the Tamiya Track and Wheel set.

Superdroid Robots
www.superdroidrobots.com

Reseller of the Johnny Robot tracks, as well as sells other Tamiya track-based kits.

Tamiya
www.tamiyausa.com

Manufacturer of plastic kits, as well as a line of educational products that include the Tamiya Track and Wheel kit, used in many third-party robots. The company offers some online sales, but at full price. You'll probably want to find a local reseller, or purchase online through a Tamiya

RC Armory sells realistic 1:8 scale remote control tanks. You can start with a basic motorized platform, and add options.

The VEX line of educational robotics lives on at Vex Labs. They offer a separate track kit that is useful for both VEX and custom robot designs.

dealer, such as Tower Hobbies (**www.towerhobbies.com** or Hobbylinc (**www.hobbylinc.com**).

VexLabs
www.vexlabs.com

Manufacturer and seller of the VEX line of robot kits. These kits were exclusively sold through RadioShack, but that company has discontinued carrying the products. The company sells a track kit that may be used with other VEX parts, or retrofitted for use on custom designs. **SV**

ABOUT THE AUTHOR

*Gordon McComb is the author of the best-selling Robot Builder's Bonanza and Electronics for Dummies. In addition to writing books, he operates a small manufacturing company dedicated to low-cost amateur robotics — **www.budgetrobotics.com**. He can be reached at robots@robotoid.com.*

Rubberbands and BALING WIRE

by Jack Buffington

Bar Codes for Robots?

How to Read a Bar Code

Barcodes are typically put onto products and are something that we usually only associate with the checkout counter. They couldn't possibly be useful for robotics, right?

Recognizing objects in its environment is not an easy task for robots these days. You can make that task much easier by sticking a bar code onto things that you want your robot to be able to recognize. You could have your robot fetch items for you simply by having it look for the proper bar code. You wouldn't even need to put the object in a specific place. It could navigate around until it saw the barcode for that object. Think about it. Wouldn't it make navigation so much easier if you could slap bar codes on the baseboards of your home and have a small database of what direction and how far to drive to get from bar code to bar code? As long as you kept the paths clear between the bar codes, your robot could get from place to place fairly easily.

It's All Black-White-Black-White

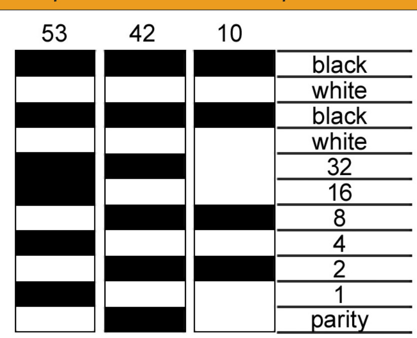
For the examples in this column, I used a special type of bar code that simplifies the process of reading them. Figure 1 shows three examples of this type of bar code. As you look at the bar codes, you'll see that they always start with four bars that are black-white-black-white. These four bars allow the program to get a good idea about how wide each bar is as it is seeing it. After that, there are six bits that are the actual data in the bar code. Finally, there is one parity bit.

You might be wondering why there are only six bits of information in the bar code. The reason is that it allows the bar code to be read at a variety of different distances. The Taos TSL3301 image sensor that was used to read the bar code could read a full eight bits, but the bar code would need to fill more of the image sensor's field-of-view. (If you are new to *SERVO Magazine*, this column covered how to read data from the Taos TSL3301 linear image sensor in the August '06 issue.) With the wide field-of-view lens that was used for this column, bar codes that used 1/4" stripes could be read from approximately four to 18 inches away. With eight bits of data, the near detection distance would have to be farther out.

Reading Bar Codes

Let's look a little more closely at how bar codes can be read. This is a bit trickier than it might initially appear.

Figure 1. Three bar codes and what they represent. Black bars represent 1s.

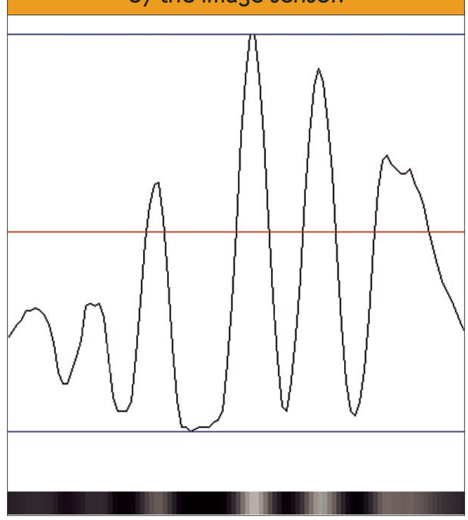


You might think that you could just aim your camera at the bar code and find the median value between the black and white stripes and simply count everything brighter than that value as a white stripe and everything darker as a black stripe. Unfortunately, that is not always possible.

Figure 2 shows a brightness graph of what a bar code could look like to the image sensor. Usually bar codes will look better than what is shown in Figure 2, but when programming, it is best to expect the worst-case situation and then make your program able to deal with it.

You might be able to tell by looking at the grayscale image at the bottom of Figure 2 that the barcode that is being viewed has a value of 52. Unfortunately, if you were to try to read it with the median value between the brightest and darkest seen values

Figure 2. A bar code as viewed by the image sensor.



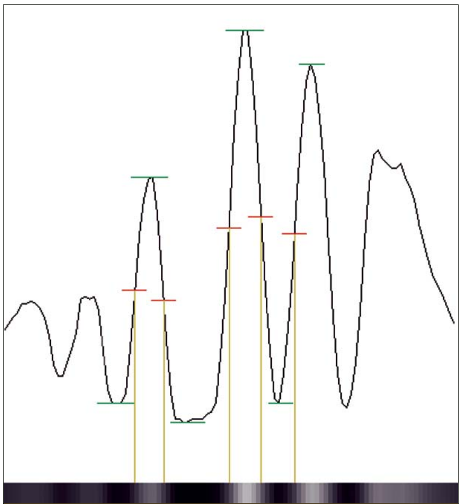


Figure 3. Graph showing the peaks, midpoints, and edges.

as shown with a red line, you wouldn't be able to read it.

In a Different Light

Because the lighting of a bar code can't be anticipated, a different way to read them is needed. A strategy that works a lot better is to find the peaks and troughs in the brightness values that the image sensor sees. You can do this by tracking the highest and lowest values as you scan through the array.

Bar codes alternate between white and black so you can make your program take advantage of the bar code's structure. When you start scanning, you will look for a minimum value at each pixel. If that pixel has a value lower than the previous minimum value, then you will make the minimum be the value of the current pixel and then you will flag that pixel as the minimum. If the value of a pixel reaches a certain amount above the minimum, then you will assume that the brightness is now becoming brighter.

In the code for this column, if the value was 20 above the minimum, it is assumed that the values are going up. Follow the maximum values, and once the values have dropped below a certain point, you will know that you have found your maximum.

At this point, you have a minimum and maximum. Find the median value between these two. Start back at the minimum value and compare each pixel to the median value. When the value passes the median value, you will have found the edge of a bar in the barcode. Repeat this process until you have scanned through all of the pixels of your image sensor.

Finding the Right Bars

Okay, so you think you've found all of the potential edges of the bar code. You actually don't know for sure that they are edges. They could be your dog, the pizza delivery man, or yesterday's socks. You'll now need to figure out what is a bar code and what isn't. To do this, we'll scan through the array looking for three bars that start with a black one and are within two pixels of each other in width. These are likely to be the beginning of a bar code.

Assuming that you orient your bar codes as shown in Figure 1 and you are using floating point math, you'll now find the distance between the top of the first black bar and the bottom of the second black bar and then divide by three. If you are running this on a microcontroller that would choke on floating point math, simply find the same distance, multiply by a power of two — such as 16 — and then divide by three. This will give you four bits of decimal precision if you multiplied by 16.

The result of either the floating point or integer calculations will give you a fairly accurate measure of the bar widths. You can now use this information to figure out if there is enough room in the image sensor to complete the bar code. If there isn't, you should keep looking for other black bars that could be a bar code.

Bar Values and Widths

Once you have found a suitable candidate to be the beginning of a bar code, you will now need to figure out what the bar values are. Initially, you might think that since you know the width of the bars you could just multiply by the bar width for every possible bar and get your result. This would work if you were using an image sensor that had more pixels and you could afford to disregard bars that were smaller than a certain number of pixels wide.

Since this sensor only has 104 pixels, you will have to test in a different way. What you will do instead is to scan through your list of edges. For each edge that you test, you will calculate the distance between that edge and the previous one. If you are using floating point math, you can simply divide this distance by the bar width that you calculated. Add .5 to the result and truncate the decimal portion by converting it into an integer. This will round the value to the nearest whole number and the result will be the number of bar widths that this stripe is wide.

If you are using integer math, you would instead find the distance between this and the previous edge and multiply that value by the same power of two as you multiplied the bar width by. Now divide that by your calculated bar width. Add eight, which is equivalent to .5 if you previously multiplied by 16. Now divide by your power of two to get things back into whole numbers. This will be how many bar widths the current stripe is wide.

Now that you know how to find the widths of the stripes, you simply need to reconstruct the number by placing 1s or 0s in a variable, depending on how many bar widths each stripe is. Do this for eight bar widths. You will need to do eight since you never counted the second white stripe and you will need to detect the parity bit, as well.

With the bars completely detected, you should take note of what the parity bit is and then divide your result register by two to remove the parity bit. The result register now contains what your program thinks is the value of the bar code.

You have one last line of defense against false detections and that is the parity bit. Count up the number of 1s in the result and if they are an even number, the parity bit should be a zero. If there is an odd number of 1s in your result, then the parity bit should be a 1, as well. If the parity bit and what you expect match, then you can be reason-

ably certain that your bar code is valid. To be fully sure though, it wouldn't hurt to move your robot a bit and see if you get the same result again.

Using the Bars

Well ... you now know how to read bar codes. Let's look at what you can do with them. One unusual — yet interesting — application is if you were trying to detect the position of a rotating shaft or linear actuator to a high level of precision. As long as it moved slowly enough, your high precision would be meaningful. What you would need to do is print bar codes onto the moving part that are spaced so that at least one complete bar code will show up in the field-of-view of the image sensor at any given position. You can identify the bar code to get a rough idea of where the object is, but you can really narrow it down by taking into consideration the pixel that sees the edge of the first black bar.

If you really want to be precise, you could take into account the gray level of the first non-white pixel of the bar code. Your position calculation would be something like bar code number * number of pixels in the sensor + number of pixels seen before the bar code * number of gray levels + gray level of first non-white pixel.

You could speed up the calculation a lot for position sensing by changing a few things. Since you can completely control the environment of the sensor, you could have only one black bar and one white bar at the beginning of the bar code since the bar width would already be known. You could ditch the parity bit, as well, since the sensor would be mounted at a fixed distance from the moving object. You could set up the lighting perfectly, so you could avoid finding peaks and detect your edges by just doing value comparisons. Reading a bar code under ideal conditions like this could turn out to be a fairly fast calculation.

Raising the Bars

As was mentioned at the beginning of this column, you could place bar codes on the floorboards of your house or apartment. This would allow your robot to navigate from bar code to bar code. That is pretty cool but consider this: If you put a very narrow field-of-view lens onto your image sensor, you could see bar codes from longer distances. With a map of X and Y coordinates for each bar code, your robot could sweep its image sensor around and figure out where multiple bar codes are and triangulate to figure out where it was in two dimensions.

It would only need to see two bar codes, but could arrive at a more accurate position using three. Using this sort of position detection, it wouldn't be restricted in where it could go in any way. Of course, finding where those bar codes are could be a tedious process. Never fear, here is an answer for that, too.

Find some retroreflective tape and draw or print your bar code onto it. If you put an ultra bright LED right next to your image sensor's lens, then the bar codes will stick out like a sore thumb. Keep in mind that this may not work at nearer distances due to the black areas becoming too bright.

If you are looking for the brightest retroreflective material on the market today, take a look at 3M's Diamond grade retroreflective materials. This material has a slight "Achilles heel" in that it stops being retroreflective when you are at too much of an angle to it. For only a slightly less retroreflective material, take a look at Scotchlite 8850, also by 3M. This is an exceedingly bright retroreflective material, as well, but can reflect light better when it is held at an angle to the light source.

Those two materials can be somewhat difficult to come by in smaller quantities, so a third final option is to go to your local automotive store and buy some retroreflective tape that is typically used to add reflectors to trailers and other equipment. This will only be about two-thirds as bright as the other two, but will likely work for you just as well.

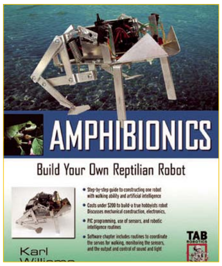
Closing the Bar

Hopefully, you have learned a thing or two about the image processing needed to detect bar codes from this column. The lowly bar code can be quite useful in certain situations. Who knows ... with some clever placement of your bar codes, you might be able to command your robot to go get you a drink from the fridge. Your bot could actually do something useful for a change. SV

Mind Candy For Today's Roboticist

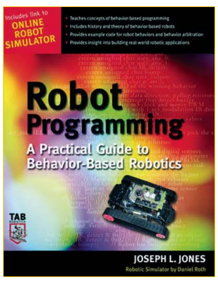
The *SERVO* Store

Amphibionics
by Karl Williams



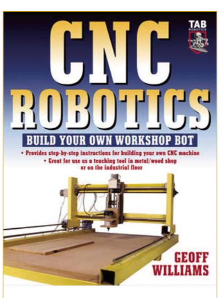
This work provides the hobbyist with detailed mechanical, electronic, and PIC microcontroller knowledge needed to build and program a snake, frog, turtle, and alligator robots. It focuses on the construction of each robot in detail, and then explores the world of slithering, jumping, swimming, and walking robots, and the artificial intelligence needed to make these movements happen with these platforms. Packed with insight and a wealth of informative illustrations. **$19.95**

Robot Programming
by Joe Jones / Daniel Roth



This ingenious book/website partnership teaches the skills you need to program a robot — and gives you a virtual robot waiting online to perform your commands and test your programming expertise. You don't need to know robotics or programming to get started! Using an intuitive method, *Robot Programming* deconstructs robot control into simple and distinct behaviors that are easy to program and debug with inexpensive microcontrollers with little memory. Once you've mastered programming your online bot, you can easily adapt your programs for use in physical robots. **$29.95**

CNC Robotics
by Geoff Williams

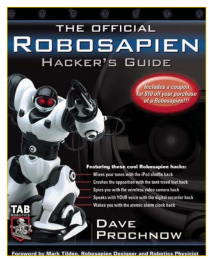


CNC Robotics gives you step-by-step, illustrated directions for designing, constructing, and testing a fully functional CNC robot that saves you 80 percent of the price of an off-the-shelf bot — and that can be customized to suit your purposes exactly, because you designed it. Written by an accomplished workshop bot designer/builder, this book gives you all the information you'll need on CNC robotics! **$34.95**

SERVO CD-Rom

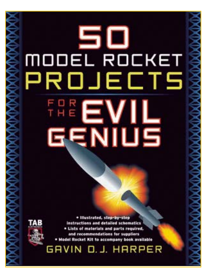
Are you ready for some good news? Along with the first 14 issues of *SERVO Magazine*, all issues from the 2005 calendar year are now available, as well. These CDs include all of Volume 1, issues 11-12, Volume 2, issues 1-12, and Volume 3, issues 1-12, for a total of 26 issues all together. These CD-ROMs are PC and Mac compatible. They require Adobe Acrobat Reader version 6 or above. Adobe Acrobat Reader version 7 is included on the discs. **$24.95 – Buy 2 or more at $19.95 each!**

The Official Robosapien Hacker's Guide
by Dave Prochnow



The Robosapien robot was one of the most popular hobbyist gifts of the 2004 holiday season, selling approximately 1.5 million units at major retail outlets. The brief manual accompanying the robot covered only basic movements and maneuvers — the robot's real power and potential remain undiscovered by most owners — until now! This timely book covers all the possible design additions, programming possibilities, and "hacks" not found anyplace else. **$24.95**

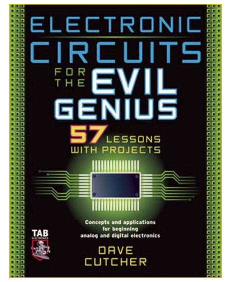
50 Model Rocket Projects for the Evil Genius
by Gavin D. J. Harper



Yes, as a matter of fact, it IS rocket science! And because this book is written for the popular Evil Genius format, it means you can learn about this fascinating and growing hobby while having run creating 50 great projects. You will find a detailed list of materials, sources for parts, schematics, and lots of clear, well-illustrated instructions. **$24.95**

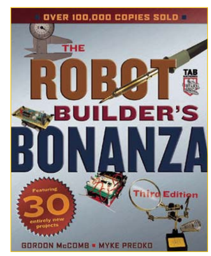
We accept VISA, MC, AMEX, and DISCOVER
Prices do not include shipping and may be subject to change.

Electronic Circuits for the Evil Genius
by Dave Cutcher



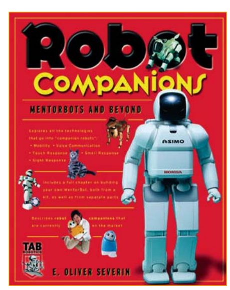
Cutcher's 57 lessons build on each other and add up to projects that are fun and practical. The reader gains valuable experience in circuit construction and design and in learning to test, modify, and observe results. Bonus website **www.books.mcgraw-hill.com/authors/cutcher** provides animations, answers to worksheet problems, links to other resources, WAV files to be used as frequency generators, and freeware to apply your PC as an oscilloscope. **$24.95**

Robot Builder's Bonanza Third Edition
by Gordon McComb / Myke Predko



Everybody's favorite amateur robotics book is bolder and better than ever — and now features the field's "grand master" Myke Predko as the new author! Author duo McComb and Predko bring their expertise to this fully-illustrated robotics "bible" to enhance the already incomparable content on how to build — and have a universe of fun — with robots. Projects vary in complexity so everyone from novices to advanced hobbyists will find something of interest. Among the many new editions, this book features 30 completely new projects! **$27.95**

Robot Companions
by E. Oliver Severin

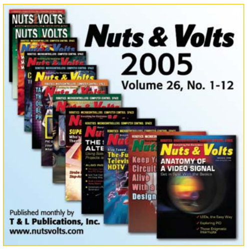


Inside *Robot Companions*, you'll find all the details, plans, and information you need to make a robot partner part of your daily life, at a price you can afford. Author E. Oliver Severin, originator of some of the technologies that make robots friendly, useful, and educational, shows you how to find or build your own robot helpmate — either from commercial kits or an assembly of separate, off-the-shelf parts. **$24.95**

To order call 1-800-783-4624 or go to our website at www.servomagazine.com

Mind Candy For Today's Roboticist

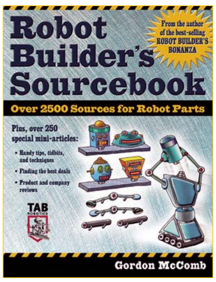
Nuts & Volts CD-Rom



Here's some good news for *Nuts & Volts* readers! Along with all 12 issues of *Nuts & Volts* from the 2004 calendar year, the 2005 issues are now available, as well. These CDs include all of Volumes 25 and 26, issues 1-12, for a total of 24 issues (12 on each CD). These CD-ROMs are PC and Mac compatible. They require Adobe Acrobat Reader version 6 or above. Adobe Acrobat Reader version 7 is included on the discs. **$24.95 – Buy 2 or more at $19.95 each!**

Robot Builder's Sourcebook

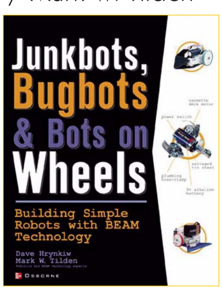
by Gordon McComb



Fascinated by the world of robotics, but don't know how to tap into the incredible amount of information available on the subject? Clueless as to locating specific information on robotics? Want the names, addresses, phone numbers, and websites of companies that can supply the exact part, plan, kit, building material, programming language, operating system, computer system, or publication you've been searching for? Turn to *Robot Builder's Sourcebook* — a unique clearinghouse of information for that will open 2,500+ new doors and spark almost as many new ideas. **$24.95**

JunkBots, Bugbots, and Bots on Wheels

by Dave Hrynkiw / Mark W. Tilden

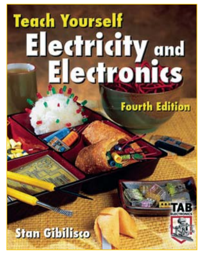


From the publishers of *BattleBots: The Official Guide* comes this do-it-yourself guide to BEAM (Biology, Electronics, Aesthetics, Mechanics) robots. They're cheap, simple, and can be built by beginners in just a few hours, with help from this expert guide complete with full-color photos. Get ready for some dumpster-diving! **$24.99**

Check out our online bookstore at www.servomagazine.com for a complete listing of all the books that are available.

Teach Yourself Electricity and Electronics — Fourth Edition

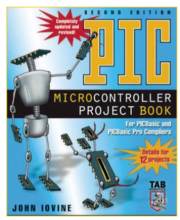
by Stan Gibilisco



Learn the hows and whys behind basic electricity, electronics, and communications without formal training. The best combination self-teaching guide, home reference, and classroom text on electricity and electronics has been updated to deliver the latest advances. Great for preparing for amateur and commercial licensing exams, this guide has been prized by thousands of students and professionals for its uniquely thorough coverage ranging from DC and AC concepts to semiconductors and integrated circuits. **$34.95**

PIC Microcontroller Project Book

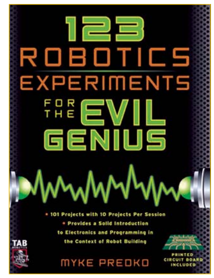
by John Iovine



The PIC microcontroller is enormously popular both in the US and abroad. The first edition of this book was a tremendous success because of that. However, in the four years that have passed since the book was first published, the electronics hobbyist market has become more sophisticated. Many users of the PIC are now comfortable shelling out the $250 for the price of the Professional version of the PIC Basic (the regular version sells for $100). This new edition is fully updated and revised to include detailed directions on using both versions of the microcontroller, with no-nonsense recommendations on which is better served in different situations. **$29.95**

123 Robotics Experiments for the Evil Genius

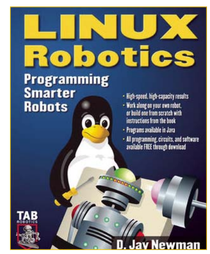
by Myke Predko



If you enjoy tinkering in your workshop and have a fascination for robotics, you'll have hours of fun working through the 123 experiments found in this innovative project book. More than just an enjoyable way to spend time, these exciting experiments also provide a solid grounding in robotics, electronics, and programming. Each experiment builds on the skills acquired in those before it so you develop a hands-on, nuts-and-bolts understanding of robotics — from the ground up. **$25.00**

Linux Robotics

by D. Jay Newman



If you want your robot to have more brains than microcontrollers can deliver — if you want a truly intelligent, high-capability robot — everything you need is right here. *Linux Robotics* gives you step-by-step directions for "Zeppo," a super-smart, single-board-powered robot that can be built by any hobbyist. You also get complete instructions for incorporating Linux single boards into your own unique robotic designs. No programming experience is required. This book includes access to all the downloadable programs you need, plus complete training in doing original programming. **$34.95**

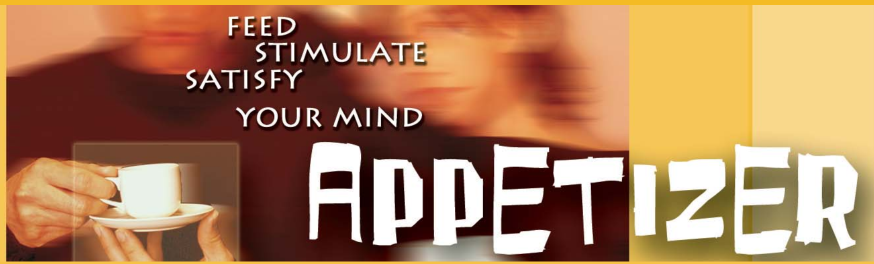
$20 each 15% off if you buy two or more.

From HomoSapien to RoboSapien

Before R2D2 there was R1D1

SERVO Magazine T-Shirts

Get your very own limited-edition *SERVO Magazine* T-shirt. Shirts come in sizes S, M, L, and XL, and are available in either black or white. All shirts are 100% pre shrunk cotton.



Hotel Earth

Nine Billion Guests and No Elevator?

by Roger G. Gilbertson

For me, space exploration and robots have always gone together hand-in-gripper. From the 1960's Surveyor robots, whose Moon landings demonstrated that future manned missions were not doomed to sink into dusty oblivion, to today's Spirit and Opportunity rovers exploring the plains and hills of Mars — the Lewis and Clarke of the Red Planet — our robots lead the way.

But don't think that we humans plan on staying behind, watching the action from armchairs. Without a doubt, everywhere our robots travel, humans will eventually follow.

As the 38th anniversary of Neil and Buzz standing on the surface of our grey neighbor approaches, more and more earthlings impatiently await their chances to fly to space. Some seek just to visit briefly, to float in zero gravity for a few minutes or days. Others have loftier desires: to establish colonies and start new societies amid the stars.And rather than just waiting and dreaming, many are doing what humans do best — picking up tools and building solutions.

Rockets and Beyond

Currently, we people of Earth send forth but a few dozen government-funded rockets each year. And, even in a good year, you can count on one hand the launches boosting people into orbit.

But as you read this, restless entrepreneurs drill holes, bend metal, and laminate exotic fibers, making components that will become the next generation of space vehicles. They work, driven by a desire to open the gates for scores — and eventually hundreds — of humans to travel where only a few have gone before.

The list of efforts inspires: SpaceShipTwo, descendent of the X Prize winning SpaceShipOne from Burt Rutan of Scaled Composites and Paul Allen of Microsoft, now joined by Richard Branson of Virgin everything; Falcon 1 and Falcon 5 from SpaceX, created by PayPal co-founder Elon Musk; Blue Origin from Jeff Bezos, founder and CEO of Amazon.com, and others.

Just far out dreaming, you say? Look up! Almost daily, crossing the sky above your home, you can glimpse the Genesis I module on its five (or more) year voyage. This prototype blazes the way for a series of inflatable space habitats that Robert Bigelow, billionaire Budget Suites hotel developer, intends to serve as floating space hotels — natural destinations for future space tourists. You can see it in the sky! The new Space Age is upon us!

LINKS

For more information about the October 2006 X Prize Cup, visit: **www.xprizecup.com/**

For more information on the NASA Space Elevator Centennial Challenge competitions, visit **www.elevator2010.org**

The Path of Growth

Government funded rockets got us started, and privately funded rockets promise to expand our opportunities (should the 900 pound NASA gorilla succeed at climbing off the table, allowing for a more level playing field for commercial space efforts). Some day, rockets will yield to an even more amazing technology capable of transporting thousands, then millions of people to and from space. Enter space elevators.

One day, a ribbon 100,000 kilometers long, as thin as plastic wrap, and many times stronger than steel will stretch from Earth to outer space. Robotic elevator cars as large as houses will climb the ribbon, taking humans and everything needed for life into Earth's orbit and beyond.

For the history and technologies for building a space elevator see "Space Elevator — Building a Highway to the Stars" in the November '05 issue of *SERVO*. In short, the 1991 discovery of carbon nanotubes — long, strong filaments of pure carbon — transformed the space elevator concept from a wild science-fiction idea into a tantalizing engineering possibility.

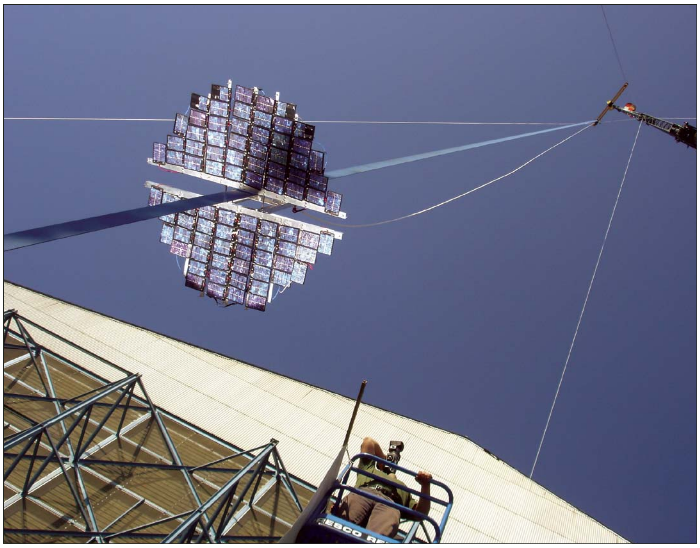
Right now, teams of engineers and experimenters labor fervently to solve the key challenges to a functional space elevator system. Slowly and steadily, humanity's largest and most spectacular engineering project departs from the station. Care to watch?

Destination: Spaceport New Mexico

On October 20 and 21, 2006, the X PRIZE Foundation and the Spaceward Foundation will present the second Space Elevator Games — a NASA Centennial Challenge — as part of the huge 2006 X PRIZE Cup events held at the Las Cruces International Airport, in Las Cruces, NM.

Against a backdrop of rocket-powered planes, roaring engine tests, and high-energy space enthusiasm, more than 20 teams have signed up to compete for $400,000 in NASA prize money. Their beam-powered climbing robots will strive to take home the cash by driving themselves hundreds of feet up into the sky on a narrow ribbon suspended from a crane. In a second event, researchers will face off in a high-tech "tractor pull" to determine who has the strongest ribbon material yet devised.

You're invited to drive down, fly in, or log on and witness the demonstrations and head-to-head competitions. Who knows, one of these ingenious, out-of-this-world designs may ultimately change the course of human history. "And you were there ..."



The University of British Columbia entry makes its way skyward during NASA's first Centennial Challenge competition held in October 2005. Photo by R. Gilbertson.

Some Thoughts on "Why?"

"Earth is the cradle of humanity, but we cannot live forever in a cradle." — Konstantin E. Tsiolkovsky, Russian astronautics pioneer, 1911

"The growth of the world population accelerated after 1900, with 2.5 billion in 1950 ... 6.1 billion in the year 2000, [and] by 2050 the world is expected to have 8.9 billion people, an increase of nearly half over the 2000 population." — United Nations Report, "The World at Six Billion," 1999

"In order to save the human race, we must develop the technologies that will allow us to live and work on other places in the Solar System ... SINGLE PLANET SPECIES DO NOT LAST and we have no idea how much time we have." — John Young, former Gemini, Apollo, and Space Shuttle astronaut, 2002

"[T]he prospect of radical life extension is only a couple of decades away ... The apparent dangers are that a dramatic reduction in the death rate will create over-population and thereby strain energy and other resources while exacerbating environmental degradation." — Ray Kurzweil, "Ray Kurzweil's Dangerous Idea" from Edge.org

Futurist Buckminster Fuller challenges us with a bold vision: "To make the world work for 100% of humanity in the shortest possible time through spontaneous cooperation without ecological offense or the disadvantage of anyone."

How do we prepare for tomorrow? How can we build and operate a management system for Spaceship Earth that meets the needs of all the guests and not just those born in the penthouse?

Tomorrow Arrives on Time

Christopher Columbus did not wait to solve Spain's domestic strife before sailing beyond the horizon. Sir Walter Raleigh did not wait to fix England's social and economic challenges before planting settlements in the New World. And we have no reason to wait to "solve" our present day problems before venturing beyond our present horizons.

Just as the Transcontinental Railway traversed the North American continent in 1869, and in doing so reduced the cost, duration, and danger of voyaging from coast to coast, so too can space elevators bridge the gap from the Earth's surface to the universe beyond.

By venturing out — first via robots and then with human explorers — by exploring, by settling new worlds and creating new societies, humanity gains unmeasurable benefits that are shared by all.

As individuals and as a species, our time on Earth is limited. By accepting these bold adventures, we do indeed risk failure, but by staying home, we guarantee it.

The universe calls, and we prepare our answer: *Ad astra!* **SV**

AUTHOR BIO

*In 1995, Roger G. Gilbertson launched **RobotStore.com**, the Internet's first commercial robotics site. In 2002, he initiated the Space Elevator Ribbon Climbing Robot Competition for the San Francisco Robot Games, featuring climbers up to 1 kilogram. After selling **RobotStore.com** to Jameco Electronics in 2005, he worked on the documentary "Who Killed the Electric Car?" from Sony Classics Pictures with directed by Chris Paine, his long time friend and business partner. Gilbertson is currently producing a documentary about the the Space Elevator. He lives in Marin County, CA, where humans still do more work than robots.*

Then and NOW

ROBOTS WHO SEE

by Tom Carroll

My last article about robots who listen got a few people who I know interested in speech recognition, as the technology is pretty much available to all of us. When it comes to robots who see, for the longest time we thought of TV and video technology and went no further. Our robots have used CdS cells and phototransistors for years, either to sense ambient light or light generated by the robot and reflected from objects. IR and other types of LEDs have been used as proximity sensors, and for line and edge detection when coupled with some sort of light sensor. But is this vision? What about robots who truly see? Do we need to refer to Issac Asimov's robots from his series of short stories — *I Robot*? Do robot eyes really glow a deep red like in his stories and many sci-fi movies? Why would eyes even *radiate* light rather than *receive* it?

The Human Eye

Human eyes are an amazing set of organs. Even without the intimate connection to the brain, the eyes accomplish a lot on their own. Add that capacity to the amazing storehouse of images that we've collected and stored in our brain throughout our lives and we can do a lot with our sight.

Vision is the one sense through which we accomplish the most difficult tasks, and the lack of sight is our greatest disability. With just a verbal description given to us, we can locate the most complex objects. Someone can tell us, "I need this special wrench that's over in that drawer. It's about 10 inches long, is bent 45 degrees in the middle, and has a ball joint on the end."

Without ever having heard of such a strange tool, much less seen one, we can quickly go over to the drawer and recognize it buried beneath many other similar wrenches, using our human vision and a keen cognitive ability. We may pick out the "ball joint," or "45-degree bend" to clue us to it. No machine vision computer can do such a thing ... *yet*. So, bypassing such complex object recognition, most of us concentrate on having our robots visualize objects, obstacles, and boundaries to guide them.

Machine Vision

In researching this article, it was quite interesting to see just how far robot vision has progressed — from TV systems and bulky computers costing hundreds of thousands of dollars, to the several ounce systems of today costing less than a hundred dollars. There's really two ways to go with robot vision (or machine vision as it is sometimes referred to in the industrial robot business).

Robots can use video means to "see" objects and transfer those images via electronic means to a TV screen that is viewed by a human. The human then makes sense of what is on the screen and reacts accordingly. Teleoperator systems use this type of technology as required by the person on the other end of the control link. This type of robot vision has been used successfully in many "robot" applications, but is no more "true vision" than a person looking at a photo album and viewing images received by a camera at some previous time.

This, however, was the only way that experimental robots of four decades ago could be controlled by visual means. These days, CCD and CMOS image sensor video cameras are quite cheap and are quite useful in teleoperated robot projects. **Supercircuits.com** has a camera for less than $12, plus many other useful types. Security Cameras Direct at **www.scdlink.com** is another vendor I have used for various types of cameras.

The other method was to have the robot sense the scene of interest and process that visual information on board in a way that was useful to the robot for object and obstacle identification, navigation, or other purposes. The "process that information" part of this description was the key. This processing required a computer and inputs from the sensor that the computer could understand.

Back in the 1960s, this was not possible with the computer technology available to any home experimenters. TV cameras used vidicon tubes and computers weighed hundreds of pounds, if not room-sized. Machine vision systems used in conjunction with industrial robots for parts identification or part orientation determination started to become available in the early 70's. Edge detection algorithms were used to isolate a particular image for processing by the computer. Two early, visual scene-processing robots — Hans

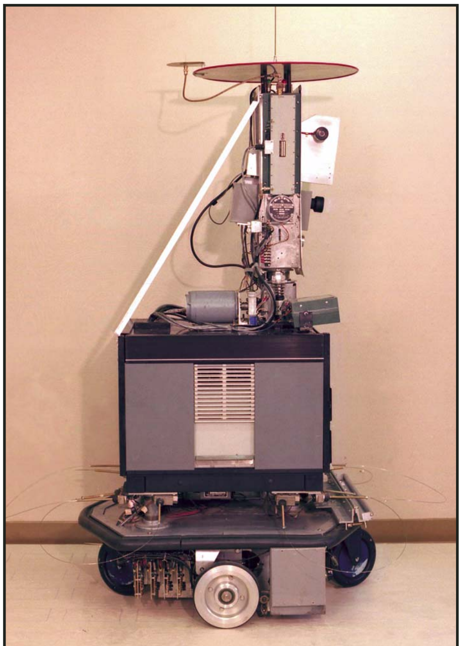


Figure 1. Shakey.

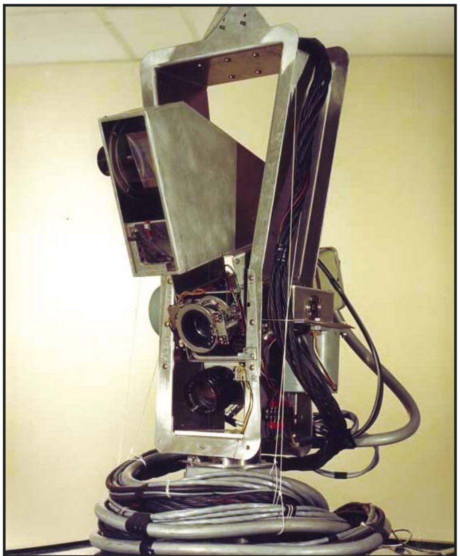


Figure 2. Shakey's cameras.

Moravec's Stanford Cart from the 1967 era was just being thought of and SRI's Shakey had yet to make the scene.

Shakey

Refer to Figure 1. I wrote a bit about Shakey in the April '04 issue of *SERVO* without mentioning very much about the sensor systems. When several articles about SRI's amazing robot made headlines in the *New York Times, Life Magazine*, and *National Geographic* in 1968 through 1970, computer science and AI really caught on with the public. *Life Magazine* referred to Shakey as the "first electronic person."

A problem-solving computer program called STRIPS controlled Shakey. The robot sensed the scene in front of it and sent the images back via a radio link to a computer that processed and analyzed the data using edge processing and then planned a course for Shakey to follow. Figure 2 shows the video yoke with several vidicon TV cameras mounted in it. After a course was plotted, Shakey moved at the incredible speed of two meters in an hour. The plotting required a synthesis of the processed scene data along with a laser range finder (helium neon) and a series of bumper sensors at its base.

In 1966, Shakey was fitted with a SDS-940 computer with 64K of 24-bit word memory — a lot for those days. Fortran and Lisp were the programs of choice. In 1969, the robot was upgraded with links to PDP-10 and PDP-15 mini computers with 192K 36-bit words of memory. The first program that was used with the new computer was STRIPS that was later replaced with LISP.

Shakey had a video system with visual scene interpretation and obstacle avoidance capabilities. It also had a planning system that both created plans and then monitored their real-world execution with a multi-level, tiered control architecture.

Shakey now resides in a glass case in the Computer History Museum in Mountain View, CA. He was also inducted into the Robot Hall of Fame at Carnegie Mellon along with a few famous real and movie robots. Shakey will be ever known as the first mobile robot that could think independently and interact with its surroundings with vision. We can only imagine that he is still looking through the glass case at all the museum's visitors and marveling at the vision technology that is available today.

These were the '60s when only a few lucky university grad students got the chance to work with intelligent robots with vision capabilities. Affordable computers were just a dream for would-be robot experimenters. Four decades later and we are blessed with powerful microprocessors and microcontrollers that can literally "run circles" around the old PDP mini-computers at less than one percent of their cost.

The Sony Aibo robot series has a vision system that can track a colored ball and allow the Aibo to interact with it by nudging it or kicking it across a floor. The Aibo also has a facial recognition feature that allows it to recognize its owner or other persons. This facial recognition type of system is a bit too complex for the average experimenter to integrate into their home-built machines, so object, obstacle, and perimeter recognition capabilities are the most desired features of a vision system.

Carnegie Mellon University and the CMUcam

I've talked a bit about some of the older vision systems and TV links, and a bit about some of the newer technologies, but let's look at a few of the systems that robot builders may be interested in. There are many robot vision system cameras available to the experimenter these days, but I'm going to discuss only two of the more popular systems. Though alumni of Harvard and Stanford Universities might dispute this, the center of basic robotics research seems to have been at Carnegie Mellon University for several decades. I was fortunate enough to be able to visit the robotics research facilities of all three of the above schools in the '80s while researching my space-borne robot design work for NASA. All of the schools, and other leading universities, excelled in specific areas of robotics, but CMU seemed to top the list in most categories, but that is strictly this author's opinion.

I was impressed with Raj Reddy, professor of Computer Science at CMU, who spent quite a bit of time with me discussing some early vision systems he was applying to his AI mobile autonomous robots. Another professor, William "Red" Whittaker, impressed me with his motto that "all robots should have practical applications." The ex-Marine who formed *CMU's Field Robotics Center* was quite distinguished in stature and was the driving force

Figure 3. H1ghlander.

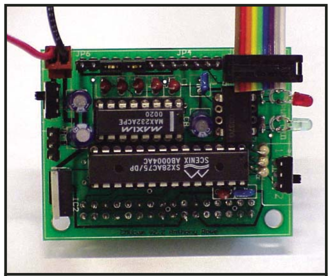


Figure 4. Back of CMUcam.

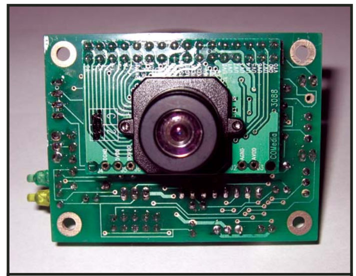


Figure 5. Front of AVRcam.

behind several amazing robots that used sophisticated vision systems, the latest being the 2005 DARPA Grand Challenge's autonomous robot vehicle named H1ghlander (Figure 3).

H1ghlander lost the race by only four minutes due to a Lidar and engine malfunction after leading most of the way — not the fault of the basic design. The H1 in the name refers to the huge red Hummer H1 (Whittaker had two in the race) that he used as the base vehicle and outfitted it with numerous vision systems and laser range finders, all tied to a GPS navigation system.

These two men typify the cutting edge technology that has emerged from CMU. It is no surprise that the CMUcam first developed at CMU has been the most popular of the vision systems available to the experimenter. We now have affordable vision systems that actually have the power within themselves to make decisions on the perceived images and act upon that information. Previous systems captured an image and sent the data to another processor to analyze the images and decide on the best action to perform.

The CMUcam was designed in 2001 by Anthony Rowe at Carnegie Mellon. The CMUcam is a low-cost, low-power vision system using the Omnivision OV6620 color CMOS imaging sensor and is designed for mobile robots. The CMUcam can do many types of on-board, real-time vision processing, thus saving precious time and memory space on a host microcontroller. The CMUcam uses a serial port and can be directly interfaced to other microcontrollers such as the PIC series and 68HC11-12s.

The CMUcam is a relatively low-powered device operating at six volts @ 200 mA of current draw, or a shade over one watt. At 17 frames per second, the CMUcam can "track the position and size of a colorful or bright object, measure the RGB or YUV (color) statistics of an image region, and automatically acquire and track the first object it sees. It can also physically track using a directly-connected servo, dump a complete image over the serial port, and dump a bitmap showing the shape of the tracked object."

Using a servo and the CMUcam, it is easy to make a robot sensor suite that pans back and forth to track an object or a wheeled robot that chases a ball around, much like the far more expensive Aibo. The CMUcam comes in a kit form or assembled at less than $100 to about $140. Figure 4 shows the back side of this amazing camera.

There is a new version of the CMU Camera — the CMUcam2 — that incorporates an on-board frame buffer that allows much more flexibility in image manipulation, sub-sampling, higher frame rates, and contains an SX52 microcontroller. This version comes completely assembled from Acroname and other suppliers for $179 or less. The following are some of the improvements of the CMUcam2 over the earlier version:

- Track user-defined color blobs at up to 50 frames per second.
- Track motion using frame differencing at 26 frames per second.
- Find the center of any tracking data.
- Gather mean color and variance data.
- Gather a 28-bit histogram of each color channel.
- Process horizontally edge filtered images.
- Transfer a real-time binary bitmap of the tracked pixels in an image.
- The camera has arbitrary image windowing.
- The camera has image down sampling.
- The camera's image properties can be adjusted.
- The user can dump a raw image (single or multiple channels).
- Up to 176 x 255 resolution is available.

Information on the CMUcam series (including the CMUcam 3) is available at **www.seattlerobotics.com** (NOT Seattle Robotics Society), Devantech, Acroname, and other suppliers.

The AVRcam Based on the Atmel Microcontroller

Another popular camera for a robot vision system is the AVRcam (see Figure 5) developed by John Orlando of the Chicago robot group — Chibots. It is also a small, low-cost, real-time image-processing engine capable of tracking colorful objects. It is based on the AVR mega8 processor from Atmel, a company that seems to be destroying itself from within these days. Despite the squabbles within the company, the Atmel series of microcontrollers have long been a favorite among robot designers and the various microcontrollers should be available for quite a while. As with the CMUcam2, the AVRcam also uses the Omnivision OV6620 imaging sensor.

The AVRcam can track eight different objects of eight different user-defined colors at 27 frames/second and can provide real-time tracked object statistics (color, bounding box, center of object, and more) through a serial port. It has an image resolution of up to 88 x 144 at 27 frames/second. It has a much lower power consumption of 5V @ 53 mA or about a quarter watt, quite a bit lower than the CMUcam.

John feels that his camera has an edge, as it does not rely on proprietary software and hardware (as does the CMUcam) and can track more objects at a higher frame rate. "One great example of this," he says, "is the international Robocup robot soccer

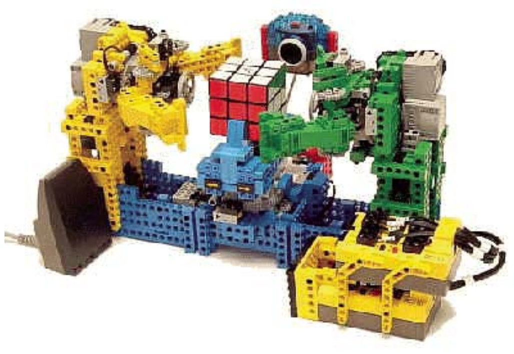


Figure 6. LEGO Cam solves Rubik's Cube.

competition held each year."

You can purchase an AVRcam from the **JRobot.net** online store, though it does require a bit of soldering or wire wrapping. The basic kit with software is $99.95.

The LEGO Cam

Another experimenter's camera for robot use is the new LEGO Cam. Logitech and LEGO Co. worked together to embed Logitech's QuickCam PC video-camera technology into LEGO Mindstorm's Vision Command product line. The new LEGO Cam with USB connection, construction elements, and advanced vision-recognition software sells for $99.

Designed to interface with the Mindstorm Robotics Invention System, using the LEGO Cam, experimenters can build and program robots that react to their surrounding environment, such as seeking and following a moving colored ball. The LEGO Cam features a 6' USB cable and has up to 30 frames per second at a respectable resolution of 352 x 288 pixels for high resolution color imaging. Though it looks a bit cheap with its bright plastic case, it does have adjustable focus and a built-in microphone. Figure 6 shows a Rubik's Cube solving "robot" using the LEGO Cam.

I've discussed three full-frame cameras available to the experimenter for robot use, yet, there is another great method that was described by *SERVO* author, Jack Buffington in the August '06 issue. It uses a 102 pixel linear array — the Taos TSL3301. It must be physically panned around a room to produce a usable image — a method that has been successfully used by many planetary explorer robot craft for years. It uses a PIC16F873 processor and the software is available on the *SERVO* website.

For those robots who live with you, it was neat enough to just provide them with a brain like the scarecrow in the *Wizard of Oz*. Now you can give them eyes. Go to the Internet and/or Google, and look up "robot vision" and you'll get millions of links. As with all of my articles, I do not and cannot go into detail on all the many types of technology available to the experimenter.

This is the last of my "Robots *Who* ..." series on those special robots that we feel most comfortable with. It is my hope that you will become interested enough to go to the many sources available and make up your own mind what is best for your robot project. **SV**

Advertiser Index